Towards Efficient and Robust Reinforcement Learning via Synthetic Environments and Offline Data

Cong Lu

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

To my parents, 陆志豹 and 姜苏.

Acknowledgements

First and foremost, I'm deeply grateful to my supervisors Mike and Yee Whye for taking a chance on me all those years ago and guiding me through the DPhil process; I will treasure the many fun discussions throughout. I want to particularly thank Mike for showing me a conscientious way to do research, the wonders of being Bayesian, and encouraging me to think about the broader implications of my research. I want to particularly thank Yee Whye for always encouraging me to formulate my thoughts scientifically and delve deeper to understand the underlying phenomena.

I am especially fortunate to have spent eight wonderful years in Oxford—one of the gems of this place is the opportunity to meet so many friendly and inspiring people. My interest in reinforcement learning began here with a master's project with Varun exploring more theoretical questions. After that, I am profoundly indebted to my many fellow students and postdocs who took the time to mentor me and introduced me to many separate fields of machine learning that I still work on today. Being able to work with such close friends throughout the DPhil has been one of the highest privileges I could have imagined.

I want to thank Tarun, Christian, and Wendelin for a fun project introducing me to deep and multi-agent RL. I want to thank Luisa for introducing me to meta-RL and also for her perspective and knowledge guiding me through the early days. I want to thank Tim for introducing me to more Bayesian and theoretical reinforcement learning, and also for endless fun discussions on maths and for always providing me with incredibly useful advice. I want to thank Phil and Jack for an awesome journey through research, which this thesis is primarily composed of—Jack particularly for his tirelessness, optimism, and vision for our ideas (also cameos of his dog), and Phil for being my partner in crime in realizing all of those ideas and making that magic happen. I also want to thank my collaborators with whom we had many fun and fruitful discussions including Leo, Max, Kristian, Sebastian, Xingchen, Robin, Vu, Guneet, Anya, Silvia, Gunshi, Karmesh, Matt, and Mikey.

I benefited a lot from two internships in industry, showing me how my research might be applied to real-world problems. I spent three lovely months at Microsoft Research

in Cambridge working on automated game testing, and I want to particularly thank Raluca, Johan, and Sam for hosting me. I also spent a great seven months working at Waymo Research in Oxford, working on simulation agents for autonomous driving. I'm particularly thankful to Max for hosting me and to Jack, João, Angad, Kyriacos, and Shimon for many useful discussions.

My DPhil was in the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems, and I'm extremely fortunate to have been funded through this program. I'm glad to have gone through the journey with my cohort; special thanks as well to Wendy, who went above and beyond to support us all. I am also thankful to Yarin and Shimon for assessing my transfer and confirmation and providing lots of advice throughout. Finally, I am deeply grateful to Professors Jakob Foerster and Tim Rocktäschel for the opportunity to discuss my work during my DPhil viva and for substantial feedback in the process.

On the personal side, I'm grateful to all the friends I made at Balliol College and beyond, far too many to list, and the fun times I had on the committee. In particular, my friends in Block C with whom we went through all the trials and tribulations of the pandemic—without you, I would have surely lost my sanity. I am grateful to my parents for their unconditional love and support. Finally, I'm extremely grateful to Anna, who has taught me more about myself than I could have ever known and who has been a constant source of kindness and support.

Originality Statement

The research described in this thesis was carried out by the author between January 2020 and October 2023 and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. The use of the first person plural is strictly a matter of style in keeping with standard scientific conventions.

Abstract

Over the past decade, Deep Reinforcement Learning (RL) has driven many advances in sequential decision-making, including remarkable applications in superhuman Go-playing, robotic control, and automated algorithm discovery. However, despite these successes, deep RL is also notoriously sample-inefficient, usually generalizes poorly to settings beyond the original environment, and can be unstable during training. Moreover, the conventional RL setting still relies on exploring and learning tabula-rasa in new environments and does not make use of pre-existing data. This thesis investigates two promising directions to address these challenges. First, we explore the use of synthetic data and environments in order to broaden an agent's experience. Second, we propose principled techniques to leverage pre-existing datasets, thereby reducing or replacing the need for costly online data collection.

The first part of the thesis focuses on the generation of synthetic data and environments to train RL agents. While there is a rich history in model-based RL of leveraging a learned dynamics model to improve sample efficiency, these methods are usually restricted to single-task settings. To overcome this limitation, we propose *Augmented World Models*, a novel approach designed for offline-to-online transfer where the test dynamics may differ from the training data. Our method augments a learned dynamics model with simple transformations that seek to capture potential changes in the physical properties of a robot, leading to more robust policies. Additionally, we train the agent with the sampled augmentation as context for test-time inference, significantly improving zero-shot generalization to novel dynamics. Going beyond commonly used forward dynamics models, we propose an alternative paradigm, *Synthetic Experience Replay*, which uses generative modeling to directly model and upsample the agent's training data distribution. Leveraging recent advances in diffusion generative models, our approach outperforms and is composable with standard data augmentation, and is particularly effective in low-data regimes. Furthermore, our method opens the door for certain RL agents to train stably with much larger networks than before.

In the second part of the thesis, we explore a complementary direction to data efficiency where we can leverage pre-existing data. While adjacent fields of machine learning, such as computer vision and natural language processing, have made

significant progress in scaling data and model size, traditional RL algorithms can find it difficult to incorporate additional data due to the need for on-policy data. We begin by investigating a principled method for incorporating expert demonstrations to accelerate online RL, KL-regularization to a behavioral prior, and identify a pathology stemming from the behavioral prior having uncalibrated uncertainties. We show that standard parameterizations of the behavioral reference policy can lead to unstable training dynamics, and propose a solution, *Non-Parametric Prior Actor–Critic*, that represents the new state-of-the-art in locomotion and dexterous manipulation tasks. Furthermore, we make advances in offline reinforcement learning, with which agents can be trained without any online data collection at all. In this domain, we elucidate the design space of offline model-based RL algorithms and highlight where prior methods use suboptimal heuristics and choices for hyperparameters. By rigorously searching through this space, we show that we can vastly improve standard algorithms and provide insights into which design choices are most important. Finally, we make progress towards extending offline RL to pixel-based environments by presenting *Vision Datasets for Deep Data-Driven RL*, the first comprehensive and publicly available evaluation suite for this field, alongside simple model-based and model-free baselines for assessing future progress in this domain.

In conclusion, this thesis represents explorations toward making RL algorithms more efficient and readily deployable in the real world. Further progress along these directions may bring us closer to the ultimate goal of more generally capable agents, that are able to both generate appropriate learning environments for themselves and bootstrap learning from vast quantities of pre-existing data.

Contents

1 Introduction

Contents

1.1 Is Reinforcement Learning a Sufficient Model of Intelligence?

One of the primary goals of modern machine learning is the development of autonomous and generally intelligent systems. Systems with such abilities could have the potential to revolutionize scientific discovery, healthcare, and engineering. Reinforcement learning (Mendel and McLaren, 1970; Waltz and Fu, 1965), which trains agents to maximize some notion of reward in an environment via interaction, has long been postulated as a solution to artificial general intelligence (Hutter, 2000, 2005; Silver et al., 2021). This belief is formalized in the *reward hypothesis* of Sutton

and Barto (2018) which states that "all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)".

Indeed, we have good reason to believe that simple reward maximization can lead to remarkably deep strategies and capabilities, as seen by many successes in superhuman Go-playing (Silver et al., 2017b), nuclear fusion control (Degrave et al., 2022), and automated algorithm discovery (Fawzi et al., 2022; Mankowitz et al., 2023). In chess, learning to maximize a single sparse reward indicating a win or loss, along with the use of self-play and Monte Carlo Tree Search (Coulom, 2006), allowed the agent *AlphaZero* (Silver et al., 2017a) to propose entirely novel insights on opening sequences, attacking techniques, and surprising long-term sacrifices (Sadler et al., 2019). Similarly, by aiming to minimize the number of assembly instructions of a target program, *AlphaDev* (Mankowitz et al., 2023) was able to automatically discover faster sorting and hashing algorithms than previous hand-designed equivalents.

Beyond training agents on a single narrow task, recent efforts have also focused on reinforcement learning in more complex and "open-ended" simulated environments (Clune, 2020; Fan et al., 2022; Küttler et al., 2020). In the *XLand* (Open Ended Learning Team et al., 2021) environment, agents are placed in a large procedurally generated 3D world with a wide variety of competitive, cooperative, and independent games. In order to generalize to unseen evaluation tasks, an agent must develop the ability to remember, reason about, and manipulate the space they reside in, as well as manage other agents in a scene. Exciting further work on this environment (Bauer et al., 2023) has shown that by combining recent advances in meta-learning, attention-based architectures, and automated curriculum learning, agents are able to adapt to new challenges as quickly as humans. A particularly exciting holy grail of open-endedness is to develop agents acting in sufficiently rich environments which are continually curious and thus exhibit *lifelong learning* (Kudithipudi et al., 2022; Wang et al., 2020).

Despite these successes, a vast amount of research is still needed to train generalist agents for the real world. Reinforcement learning agents are often exceptionally sample-inefficient (Yu, 2018), requiring billions of interactions to learn tasks from scratch that humans can perform in a matter of hours. Moreover, they are often brittle to even slight changes in environment (Henderson et al., 2018; Zhang et al., 2018) including dynamics, observations, or goals. This is in stark contrast to biological agents that are equipped with strong inductive biases and priors (Martin, 2007; Spelke and Kinzler, 2007) for reasoning about our world that make us vastly more efficient and robust. For example, learning to cook a dish in one's home kitchen would likely mean that one could cook the same dish in a vastly different kitchen; this remains an insurmountable challenge for current agents.

There are also methods of acquiring knowledge that clearly lie outside the traditional reinforcement learning paradigm in biological systems, such as innate knowledge. Many animals including horses, gazelles, and fish are able to move in a well-coordinated manner shortly after birth. This is critical for survival as these animals often live in environments with natural predators (Walther, 1969); however, these abilities are more in-built rather than learned. Similarly, in humans, an innate aversion to certain smells from birth encourages us to avoid spoiled food without the need to try it and receive direct feedback (Darwin et al., 1998). An alternative paradigm of biological learning known to be mechanistically distinct from individual learning is observational learning (Bandura, 2008; Isbaine et al., 2015). The machine learning equivalent, learning from demonstration (Argall et al., 2009; Schaal, 1996), has also proven successful using data without reward.

Furthermore, while Silver et al. (2021); Sutton and Barto (2018) postulate that simple reward maximization is sufficient for general intelligence, it is often difficult in practice to accurately specify such a reward function to optimize against (Clark and Amodei, 2016; Pan et al., 2022). A prominent example of this is autonomous driving, where an agent is trained to drive a vehicle towards a pre-specified destination, but must also do so safely in accordance with human norms and traffic rules. Critically,

in this case, it is not only important *what* the agent does but also *how* it does it. This is further complicated by the fact that humans often disagree on the right response in a situation. As such, many proposed reward functions are often extensively hand-crafted and shaped based on empirical performance (Kiran et al., 2021). Even then, Knox et al. (2023) demonstrated that across 9 publicly available proposed reward functions for autonomous driving, the best would result in a policy that crashes 4000 times as often as a drunk 16–17 year old.

As we contend with these unresolved issues, recent developments in supervised learning have shown the potential of a new paradigm driven by rampant data and model capacity scaling (Brown et al., 2020; Chowdhery et al., 2022). This has resulted in a wide range of powerful generative language (OpenAI, 2023; Touvron et al., 2023) and image "foundation" models (Rombach et al., 2022) becoming available for popular consumer use, and has begun to have repercussions in the field of reinforcement learning. Rather than learning from scratch, Wang et al. (2023) proposed *Voyager*, a gradient-free agent that continually explores the world and acquires diverse skills entirely in-context (Akyürek et al., 2022), with GPT-4 (OpenAI, 2023). Voyager acts by executing code in the environment and generates a library of useful skills that may be composed together for more complex behavior. Suitable tasks are generated by GPT-4 in order to maximize novelty based on the agent's current state and capabilities. Remarkably, this agent is able to quickly explore the in-game world, progress rapidly through the tech tree, and even leverage learned skills in a new instance of the game.

These findings force us to re-consider what the exact role of reinforcement learning should be in a generally intelligent system. Since reinforcement learning remains the most effective method for discovering general superhuman skills and exploring entirely unknown environments, a particularly exciting path forward could be to incorporate insights from the scaling revolution back into reinforcement learning. Nair et al. (2022); Szot et al. (2023) demonstrate how pre-trained language and visual representations from internet-scale data can be used to vastly improve generalization

in embodied robotics tasks. On the other hand, video foundation models (Hu et al., 2023) have recently made great strides in being able to simulate the world with high fidelity and present us with the opportunity to train real-world agents with far less interaction with the environment. In the next section, we lay the groundwork for these ideas and take a data-centric view of two key open questions in reinforcement learning.

1.2 The Problems of Data Efficiency and Robustness

In this thesis, we focus on addressing two of the central shortcomings of modern reinforcement learning, by expanding or modifying the training data an agent sees. In particular, we develop methods for improving efficiency and robustness using synthetic model-based algorithms (Sutton, 1991) and offline data (Ernst et al., 2005; Levine et al., 2020). We begin by highlighting how resolving these issues would play a pivotal role in the real-world deployability of reinforcement learning agents.

Data Efficiency. As the data requirements for reinforcement learning are often vast, many of the successes in this field to date have been in simulated environments that can be run faster than real-time, effectively parallelized, and easily reset to initial conditions (Haarnoja et al., 2018; Mnih et al., 2015; Silver et al., 2017b; Vinyals et al., 2019). In stark contrast, real robot hours are expensive, and typical random exploration strategies may lead to unsafe behavior and damage to the physical system (Dulac-Arnold et al., 2019). As a result, these robots also often need manual human intervention to either reset to initial conditions or prevent unsafe actions. The complementary fields of offline reinforcement learning and model-based reinforcement learning were precisely designed to reduce the quantity of data required to train agents.

Offline reinforcement learning offers an alternate approach to training policies by leveraging pre-existing data, thereby circumventing the need for online data collection. This approach effectively addresses the challenges associated with unsafe

tabula-rasa exploration in a new environment. Moreover, the learned offline policies can be further refined and improved through a small number of online steps. In parallel, model-based reinforcement learning seeks to maximize the utility of available data by learning a simulator of the environment, enabling the generation of synthetic rollouts for policy training. Both of these research domains have yielded significant advancements in enhancing the deployability of reinforcement learning methods in real-world applications (Lutter et al., 2021; Wu et al., 2022).

Robustness. As we look to deploying reinforcement learning agents in the real world, they must be robust to changes to the environment as well as stable to train. Leveraging learned models to train agents not only improves data efficiency but also provides the opportunity to improve generalization from existing data by allowing agents to take actions in previously unseen states (Young et al., 2023). Kirchmeyer et al. (2022); Lee et al. (2020) take this even further and show by equipping dynamics models with a "context" or descriptor of an environment, they are able to generalize to unseen new tasks, which in turn leads to agents to generalize better downstream.

Conversely, the utilization of prior data, particularly high-quality expert demonstrations, can significantly contribute to stabilizing the training process. In hard exploration environments with sparse reward signals, finding any positive reward can be challenging which can lead agents to have high variability during training (Salimans and Chen, 2018). However, the availability of expert demonstrations can offer agents access to an alternate reliable training signal, leading to improved consistency and performance (Hester et al., 2017; Nair et al., 2018). Even without expert-level data, Ball et al. (2023) show that mixing in suboptimal prior data in a balanced way can also help stabilize and accelerate existing algorithms.

1.3 Thesis Structure and Contributions

The remainder of the thesis is structured as follows. We begin with a literature review covering the necessary prerequisites and background on online and offline

reinforcement learning, and the algorithms used in subsequent chapters. The thesis is then divided into two parts. The first part furthers the development of synthetic environments and data for reinforcement learning, and the second part makes advances in reinforcement learning from offline data. As this is an integrated thesis, each subsequent chapter corresponds to a published conference paper or journal paper that I led or co-led. At the beginning of each chapter, we introduce the problem setting and approach in relation to the overall thesis and highlight the key contributions. We include the appendices of each chapter at the end of the thesis.

1.3.1 Part I: Synthetic Environments and Data for Reinforcement Learning

In this part, we make novel contributions to the development of synthetic learning environments that allow agents to generalize successfully to unseen tasks and train robustly from less data.

- **Chapter 3** proposes *Augmented World Models*, an algorithm designed for offline-to-online transfer where the dynamics at test-time may differ from the training data. Our method augments a learned dynamics model with simple transformations that seek to capture potential changes in the physical properties of a robot, leading to more robust policies. Furthermore, this defines a training distribution of tasks in the meta-learning sense which allows us to train a contextual policy and infer the context at test time. Combined, this leads to a powerful approach to enhancing an agent's ability to generalize zero-shot to novel dynamics on a variety of simulated locomotion tasks. This chapter is based on the following publication (Ball et al., 2021): *Philip J. Ball*, **Cong Lu***, Jack Parker-Holder, and Stephen J. Roberts. Augmented World Models Facilitate Zero-Shot Dynamics Generalization From a Single Offline Environment.* ***In ICML, 2021.***[1]

[1]Here and throughout, an asterisk (*) indicates joint-first authorship.

Contributions: This project was jointly conceived by the first three authors, who also wrote the paper. Philip J. Ball led the offline analysis and implementation of the baseline. Cong Lu led the implementation and evaluation of the core algorithm. Jack Parker-Holder contributed to the ideas and design. Stephen J. Roberts contributed to the supervision of the project.

- **Chapter 4** develops a novel paradigm for model-based reinforcement learning, namely using generative modeling to directly model and upsample the training data distribution of an agent. Our approach, *Synthetic Experience Replay*, leverages recent advances in diffusion generative models to upsample data for reinforcement learning. We demonstrate that generative training data offers superior performance compared to, and also seamlessly integrated with standard data augmentation. Most notably, our method also opens the door for certain reinforcement learning agents to train stably with much larger networks. This chapter is based on the following publication (Lu et al., 2023): *Cong Lu*, Philip J. Ball*, Yee Whye Teh, and Jack Parker-Holder. Synthetic Experience Replay.* ***In NeurIPS, 2023.***

  Contributions: Cong Lu proposed the project and led the design of the diffusion model, offline experiments, and paper writing. Philip J. Ball led the online experiments and the analysis of our algorithm, and contributed to the paper writing and design of the diffusion model. Yee Whye Teh contributed to the supervision of the project. Jack Parker-Holder contributed to the supervision of the project, experimental design and paper writing.

1.3.2 Part II: Reinforcement Learning from Offline Data

In the second part, we develop principled methods to better utilize prior data in reinforcement learning. This includes integrating expert demonstrations into online reinforcement learning, fundamental methodological contributions to model-based offline reinforcement learning, and the development of principled benchmarks and baselines for offline reinforcement learning from visual observations.

- **Chapter 5** begins by showing that Kullback Leibler-regularized reinforcement learning with behavioral reference policies derived from expert demonstrations can suffer from pathological training dynamics due to poor predictive uncertainties in the prior. In fixing this pathology, we develop *Non-Parametric Prior Actor–Critic*, which results in state-of-the-art performance on locomotion and dexterous manipulation tasks. This chapter is based on the following publication (Rudner et al., 2021): *Tim G. J. Rudner*, **Cong Lu***, Michael A. Osborne, Yarin Gal, and Yee Whye Teh. On Pathologies in KL-Regularized Reinforcement Learning from Expert Demonstrations.* ***In NeurIPS, 2021.***

  Contributions: Tim G. J. Rudner conceived the project and led the theoretical foundations and paper writing. Cong Lu led the experiments and empirical evaluation of the algorithm and contributed to the paper writing. Michael A. Osborne, Yarin Gal, and Yee Whye Teh contributed to the supervision of the project.

- **Chapter 6** examines the gaps between theory and empirical practice in offline model-based reinforcement learning and shows that prior methods which rely on estimating dynamics uncertainty often choose sub-optimal heuristics for doing so. By leveraging best practices from supervised learning, along with thorough tuning of the associated hyperparameters, we can vastly improve standard algorithms and provide insights into which design choices are most important. This chapter is based on the following publication (Lu et al., 2022a): *Cong Lu*, Philip J. Ball*, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts. Revisiting Design Choices in Offline Model Based Reinforcement Learning.* ***In ICLR (Spotlight), 2022.***

  Contributions: This project was jointly conceived by the first three authors, who also wrote the paper. Cong Lu led the empirical evaluation and implementation of the Bayesian Optimization agent. Philip J. Ball led the analysis of design choices and implementation of the uncertainty penalties.

Jack Parker-Holder contributed to the ideas and design. Michael A. Osborne and Stephen J. Roberts contributed to the supervision of the project.

- **Chapter 7** develops the first comprehensive and publicly available evaluation suite for the nascent field of offline reinforcement learning in high-dimensional pixel-based environments, *Vision Datasets for Deep Data-Driven RL*, alongside identifying key desiderata that algorithms should satisfy. Alongside the benchmarks, we establish simple model-based and model-free baselines for the aforementioned desiderata to serve as a measure of progress for future advances in this domain. This chapter is based on the following publication (Lu et al., 2022b): *Cong Lu*, Philip J. Ball*, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and Opportunities in Offline Reinforcement Learning from Visual Observations.* ***In TMLR, 2023.***

  Contributions: This project was jointly conceived by the first four authors, who also wrote the paper. Cong Lu led the implementation of the model-based algorithms and overall empirical evaluation. Philip J. Ball led the creation of the benchmark and implementation of the model-free algorithms. Tim G. J. Rudner and Jack Parker-Holder contributed to the ideas and high-level design. Michael A. Osborne and Yee Whye Teh contributed to the supervision of the project.

Finally, we conclude the thesis by reflecting on the work in the context of current developments in artificial intelligence and theorizing what a roadmap to efficient and robust agents may look like in the future.

2

Background

Contents

In this chapter, we introduce the necessary prerequisites and key concepts for the thesis. The content follows the overall structure of the thesis and begins with an overview of reinforcement learning. Next, we present an introduction to model-based methods, which is the focus of Part I. Finally, we present offline reinforcement learning, which is the focus of Part II. In the subsequent chapters, further background will be introduced as needed.

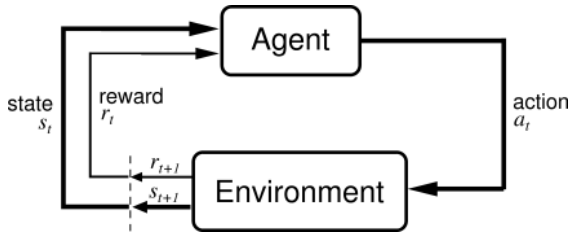


Figure 2.1: The interaction between an agent and its environment in reinforcement learning. At each timestep t, the agent observes the current state s_t, chooses an action a_t, and receives a reward r_t. The environment then transitions to a new state s_{t+1}.

2.1 Reinforcement Learning

We begin by introducing the reinforcement learning framework and model-free solution methods. We additionally introduce two important subproblems in reinforcement learning we investigate in the thesis: meta-reinforcement learning where one must learn to generalize across different tasks and automated hyperparameter tuning for reinforcement learning.

2.1.1 Markov Decision Processes

We make the standard assumption that our environment takes the form of a Markov Decision Process (MDP, Sutton and Barto (2018)), defined as a 5-tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in (0, 1)$ the discount factor. The goal in reinforcement learning is to optimize a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return

$$J(\pi) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]. \tag{2.1}$$

Interaction with the environment proceeds in discrete time steps $t = 0, 1, 2, \ldots$ and so on. At each time step t, the agent observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to its policy π, and receives a reward $r_t = R(s_t, a_t)$. The environment then transitions to a new state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. We illustrate this in Figure 2.1. The agent typically begins with no knowledge of the environment and must learn to efficiently trade off exploration and exploitation.

Example: Go playing. As a motivating example, consider the setting in AlphaGo (Silver et al., 2017b) where an agent is trained to play the game of Go on a 19x19 board against an opponent. The state space $\mathcal{S}$ is the set of all legal board configurations and the action space $\mathcal{A}$ is discrete containing the moves: play at any location on the board, resign, or pass. The reward function R is sparse: 1 if the agent wins the game, -1 if the opponent wins, and 0 otherwise. The transition dynamics $P(s' \mid s, a)$ are given by the rules of the game and the opponent's actions.

2.1.2 Value Functions and Bellman Equations

In order to learn an optimal policy for a given environment, it is useful to define the notion of a value function. The state-value function $V^\pi(s)$ is defined as the expected discounted return starting from state s and following policy π thereafter

$$V^\pi(s) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right]. \tag{2.2}$$

Similarly, we define the action-value function $Q^\pi(s, a)$ as the expected discounted return starting from state s, taking action a, and following policy π thereafter

$$Q^\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a\right]. \tag{2.3}$$

These value functions satisfy the Bellman equations (Schweitzer and Seidmann, 1985)

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s, a)\right] \tag{2.4}$$

$$Q^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right]. \tag{2.5}$$

We next define an optimal policy π^* as one that maximizes the expected discounted return from all states, i.e.

$$\pi^* = \arg\max_\pi V^\pi(s). \tag{2.6}$$

While the policy may not be unique, all optimal policies share the same value functions, which we denote $V^*(s)$ and $Q^*(s, a)$. The optimal value functions satisfy

the Bellman optimality equations

$$V^*(s) = \max_a Q^*(s,a) \tag{2.7}$$

$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^*(s') \right]. \tag{2.8}$$

Given the optimal value functions, the optimal policy may be recovered as

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} Q^*(s,a') \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

2.1.3 Model-Free Reinforcement Learning

From here on, we assume that we are in the "deep" reinforcement learning setting (Baird, 1995; Precup et al., 2001; Sutton et al., 1999), where the state and action spaces may be very large or continuous. In order to deal with these large spaces, we typically use function approximation to represent the policy and value functions. The standard choice is to use deep neural networks (LeCun et al., 2015), which means that a policy $\pi_\theta(a|s)$ would be parameterized by a network with parameters θ. We introduce three canonical model-free reinforcement learning paradigms: value-based, policy-gradient, and actor-critic methods together with typical algorithms for each.

Value-Based Methods. Value-based methods (Watkins and Dayan, 1992) attempt to learn the optimal value function $V^*(s)$ or $Q^*(s,a)$ from data. The canonical algorithm for large state spaces with finite actions is Deep Q-Networks (DQN, Mnih et al. (2015)). This was the first algorithm to successfully learn control policies directly from high-dimensional sensory inputs using reinforcement learning and surpassed a human expert across the Atari 2600 suite. In DQN, the optimal action-value function $Q^*(s,a)$ is represented by a neural network $Q_\theta(s,a)$ with parameters θ. The parameters of the Q-network are learned by minimizing the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\left(Q_\theta(s,a) - y \right)^2 \right], \tag{2.10}$$

where $y = r + \gamma \max_{a'} Q_{\theta^-}(s',a')$ is the target value derived from the Bellman equation and θ^- are the parameters of a "target network" that is periodically

updated to match the current network. This is necessary to avoid issues of circularity when using function approximation, as naïvely updating both the target and current networks can lead to divergence. Interactions with the environment are stored in a replay buffer $\mathcal{D}$ to facilitate data reuse, and the loss is minimized by stochastic gradient descent.

In order to balance exploration and exploitation, DQN uses an ϵ-greedy policy to select actions. This means at time step t, the agent takes the action

$$a_t = \begin{cases} \arg\max_a Q_\theta(s_t, a) & \text{with probability } 1 - \epsilon \\ \text{random} & \text{with probability } \epsilon. \end{cases} \tag{2.11}$$

The value of ϵ typically begins at 1 and is annealed to a small value over the course of training. Since the value function learned is that of the optimal policy which differs from the exploratory policy, DQN is described as an "off-policy" algorithm.

Policy Gradient Methods. Next, we introduce policy gradient methods that aim to learn the optimal policy $\pi^*(a|s)$ directly. These methods are more directly applicable in cases where the action space is continuous, such as in robotic control. The policy gradient theorem (Sutton et al., 1999) states that the gradient of the expected return with respect to the policy parameters is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a) \right]. \tag{2.12}$$

The value function may be estimated by Monte Carlo rollouts, or using a learned value function. The policy parameters are then updated by stochastic gradient ascent. Vanilla policy gradient algorithms typically suffer from instability due to high variance in the gradient estimates. To remedy this, a variety of methods have been proposed to stabilize training, including Trust Region Policy Optimization (TRPO, Schulman et al. (2015)) and Proximal Policy Optimization (PPO, Schulman et al. (2017b)) which focus on constraining the magnitude of the policy updates to avoid catastrophic drops in performance. As these algorithms are based on the return of the current policy, they are described as "on-policy".

Actor-Critic Methods. Finally, we introduce actor-critic (Greensmith et al., 2004; Konda and Tsitsiklis, 1999) methods, which are a family of methods that aim to combine the strengths of both value-based and policy gradient methods. We focus on the Soft Actor-Critic (SAC, Haarnoja et al. (2018)) algorithm, an off-policy actor-critic algorithm which is the predominant algorithm used in this thesis (in Chapters 3, 4, 6) This algorithm was designed to both address the common instabilities found in deep off-policy value-based algorithms (van Hasselt et al., 2018) and alleviate the requirement for on-policy data in policy gradient methods which prevents data reuse. SAC is based on maximum entropy reinforcement learning (Ziebart et al., 2008) which augments the return with an entropy term that encourages the agent to achieve high return whilst behaving as randomly as possible. The objective function is defined as

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot|s_t\right)\right)\right)\right], \tag{2.13}$$

where $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state s_t and α is a temperature parameter.

The policy is represented by a neural network $\pi_\phi(a|s)$ with parameters ϕ and the value function is represented by a neural network $Q_\theta(s, a)$ with parameters θ. SAC follows the standard actor-critic format where training alternates between a policy evaluation step and a policy improvement step. The policy evaluation step updates the value function by minimizing the Bellman error similar to Equation (2.10) with Double Q-Learning (van Hasselt et al., 2016). The policy improvement step updates the policy towards the exponential of the Q-function by minimizing the KL according to

$$\pi_{\phi'} = \arg\min_{\pi_{\phi'}} \mathbb{E}_{s\sim\mathcal{D}}\left[\mathrm{KL}\left(\pi_{\phi'}(\cdot|s)||\frac{\exp(Q_\theta(s,\cdot))}{Z_\theta(s)}\right)\right], \tag{2.14}$$

where $Z_\theta(s) = \int_a \exp(Q_\theta(s, a))\mathrm{d}a$ is the partition function which normalizes the distribution.

2.1.4 Meta Reinforcement Learning

So far, we have only discussed reinforcement learning in the context of a single fixed environment. An important setting that we consider in Chapter 3, is meta-reinforcement learning (Finn et al., 2017; Rakelly et al., 2019a; Wang et al., 2016; Zintgraf et al., 2021a,b) where the agent may be deployed over a variety of different related environments. Formally, the meta-RL setup consists of a distribution over MDPs $p(M)$ from which we can sample during meta-training. Each MDP $M_i \sim p(M)$ is defined by a tuple $M_i = (\mathcal{S}, \mathcal{A}, P_i, R_i, \gamma)$—i.e. the tasks share the same state and action space but the transition dynamics and reward function may change. The index i represents an unknown task description, which could be a goal position or identifier. Sampling an MDP from $p(M)$ is typically done by sampling from a distribution over dynamics $p(P, R)$. At meta test-time, the agent is evaluated based on the average return over tasks from the distribution. Achieving high performance in this setting involves both generalizing across related tasks and trading off exploration-exploitation when reasoning about task uncertainty.

2.1.5 Hyperparameter Tuning

A crucial factor limiting the deployment of reinforcement learning to new problems is the notorious sensitivity of algorithms with respect to their hyperparameters (Andrychowicz et al., 2021; Engstrom et al., 2020; Henderson et al., 2018), which often require expensive tuning. Indeed, it has been shown that when tuned effectively, good configurations often lead to dramatically improved performance in large-scale settings over the default (Chen et al., 2018). A powerful method for automated hyperparameter tuning which we use in Chapter 6 is Bayesian Optimization (BO, Brochu et al. (2010); Garnett (2023); Jones et al. (1998); Shahriari et al. (2016)).

Bayesian Optimization. Bayesian Optimization is a sequential model-based optimization method that aims to find the global optimum of some $f : \mathcal{X} \to \mathbb{R}$ which may not admit gradient observations. The function f is assumed to be expensive

to evaluate, and we are only allowed to query it at a finite number of points. In our case, this makes BO particularly applicable to hyperparameter tuning, where the function f is the final performance of an RL agent. The two key components of Bayesian optimization are a probabilistic surrogate model of the objective function and an acquisition function that determines the next point to query.

Typically, the surrogate model is a Gaussian Process (GP, Williams and Rasmussen (2006)), a flexible class of non-parametric models that can be used to represent a distribution over functions f under the assumption that any finite subset $\{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^n$ is jointly Gaussian. We additionally assume that we observe noisy evaluations of the function f, i.e. $y_i = f(\mathbf{x}_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. A GP is then defined by a mean function $m(\mathbf{x})$ (usually zero) and a covariance function $k(\mathbf{x}, \mathbf{x}')$ which is a similarity measure between two points that encodes our prior belief about the function. Given a set of observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we may perform exact Bayesian inference to compute the posterior distribution over functions $p(f|\mathcal{D})$. For a new point $\mathbf{x}_*$, the predictive distribution $p(y_*|\mathbf{x}_*, \mathcal{D})$ is also Gaussian with mean and variance given by

$$\mu(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x})(K + \sigma^2 I)^{-1}\mathbf{y}, \tag{2.15}$$

$$\mathrm{Var}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})(K + \sigma^2 I)^{-1}k(\mathbf{x}, \mathbf{x}_*) + \sigma^2, \tag{2.16}$$

where K is the $n \times n$ matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{y} = [y_1, \ldots, y_n]^\top$.

Next, an acquisition function is derived from the surrogate model to determine the next point to query. A common choice here is Upper Confidence Bound (UCB, Srinivas et al. (2010)), which is defined at each timestep t as

$$\alpha_{\mathrm{UCB}}(\mathbf{x}_t) = \mu(\mathbf{x}_t) + \beta_t \mathrm{Var}(\mathbf{x}_t), \tag{2.17}$$

where β_t is a hyperparameter that controls the trade-off between exploration and exploitation. At each BO iteration, we select samples that maximize the acquisition function and then update the surrogate model with the new observations. This process is repeated until the evaluation budget is exhausted, and the global optimum

is estimated based on the collected data. As reinforcement learning hyperparameters are often mixed-modality (e.g. continuous and discrete) and high-dimensional, we use the algorithm Casmopolitan (Wan et al., 2021). Casmopolitan is a state-of-the-art BO method designed for mixed modalities that has already been used for automated hyperparameter tuning elsewhere in reinforcement learning (Wan et al., 2022).

2.2 Part I: Model-Based Reinforcement Learning

In the previous section, we focused on model-free algorithms that directly optimize a policy or value function using real experience from the environment. However, this can be sample inefficient, and a complementary approach to traditional model-free reinforcement learning is to learn a model of the environment and use this to generate synthetic data for training a policy. Recent model-based methods (Hafner et al., 2020a,b; Janner et al., 2019) have shown large speedups relative to model-free equivalents across a variety of control tasks with methods such as DayDreamer (Wu et al., 2022) learning to control a real robot in a single hour.

On a conceptual level, we may ask why generating synthetic data makes sense, especially in light of the data-processing inequality (Beaudry and Renner, 2012) which states that "post-processing cannot increase information" and we are in this sense bottlenecked by real data. One explanation for the success of model-based methods is that generalization is often easier in supervised learning compared to reinforcement learning, due to the non-stationarities in the learning process (Igl et al., 2021). As such, learning supervised models can make particular sense to make the most out of the available data. Furthermore, depending on the environment, an optimal policy or value function could be complex and harder to learn than the dynamics. In this section, we introduce model-based reinforcement learning, latent models for high-dimensional observation spaces, and finally an introduction to generative modeling.

2.2.1 Dyna-Style Methods

A generic and unifying framework for model-based reinforcement learning is given by Dyna (Sutton, 1991), which interleaves model learning and planning with real experience. Recent realizations of this often use learned forward dynamics models (Hafner et al., 2020a,b; Janner et al., 2019). Concretely, given a dataset of transitions $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$, we can learn an approximate dynamics model $P_\psi(s_{t+1}, r_t | s_t, a_t)$. Given this learned model, we can generate rollouts from any given state s by sampling actions via some exploratory policy and then generating the next state s' and reward r from the model $s', r \sim P_\psi(\cdot|s, a)$. These model rollouts can then be concatenated with real experience and used to update the policy or value function.

A modern approach that we build on in Chapters 3 and 6 is Model-Based Policy Optimization (MBPO, Janner et al. (2019)), which uses deep ensembles (Lakshminarayanan et al., 2017) to represent the dynamics model and Soft Actor-Critic (SAC, Haarnoja et al. (2018)) as the policy optimizer. Each model is trained via maximum likelihood estimation, where the model parameters ψ are updated by maximizing the log-likelihood of the data

$$\psi^* = \arg\max_{\psi} \sum_{t=1}^{T} \log P_\psi(s_{t+1}, r_t | s_t, a_t). \tag{2.18}$$

The ensemble is thus a collection $\{P_\psi^{(i)}\}_{i=1}^{B}$ of B neural networks and to generate a prediction, a model is selected at random. The random model selection allows for different transitions along a single rollout to be generated from different individual models. Where data is scarce and we have high epistemic uncertainty, the ensemble is likely to have high variance in the generated next state and reward, which prevents model exploitation. Furthermore, MBPO showed in theory and in practice that using short model-generated rollouts branched from real data (e.g. with a rollout horizon $k \approx 20$) is sufficient to reap the benefits of model-based RL while avoiding issues of compounding error with full-length rollouts.

2.2.2 Latent Dynamics Models

As we move beyond low-dimensional proprioceptive inputs for environments, modeling the environment dynamics directly in large observation spaces becomes intractable. In Chapter 7, we consider a different approach suitable for pixel-based environments using latent dynamics models, DreamerV2 (Hafner et al., 2020b). DreamerV2 was the first reinforcement learning agent to achieve human-level performance on Atari games solely using synthetic rollouts, learning a model of the environment using a Recurrent State Space Model (RSSM, Hafner et al. (2019, 2020a). Given a sequence of images $x_{1:T}$, actions $a_{1:T}$, and rewards $r_{1:T}$, the RSSM learns model states s_t containing a deterministic component h_t, implemented as the recurrent state of a Gated Recurrent Unit (GRU, Chung et al. (2014)), and a stochastic component z_t with a categorical distribution. This representation is additionally trained with a reconstruction loss to the original image. Similarly to MBPO in Section 2.2.1, the actor-critic policy in DreamerV2 is trained on short imagined trajectories of latent states branched from real data.

2.2.3 Generative Modeling

To conclude this section, we introduce the prerequisites for Chapter 4 which proposes a novel model-based approach to generating synthetic data for training a reinforcement learning agent via generative modeling. Generative modeling is a subfield of machine learning concerned with directly modeling the data distribution $p(\mathbf{x})$. This has traditionally been applied in image generation (Goodfellow et al., 2014; Kingma and Welling, 2014), where the goal is to learn a model that can produce realistic images based on a dataset. In this thesis, we consider modeling the reinforcement learning data distribution and particularly focus on diffusion generative models, a recent class of models that have shown remarkable sample quality and diversity whilst remaining stable to train.

Diffusion Models. Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015) are a class of generative models inspired by non-equilibrium thermodynamics that learn to iteratively reverse a forward noising process and generate samples from noise. Given a data distribution $p(\mathbf{x})$ with standard deviation σ_{data}, we consider noised distributions $p(\mathbf{x};\sigma)$ obtained by adding i.i.d. Gaussian noise of standard deviation σ to the base distribution. The forward process is defined by a sequence of noised distributions following a fixed noise schedule $\sigma_0 = \sigma_{\max} > \sigma_1 > \cdots > \sigma_N = 0$. When $\sigma_{\max} \gg \sigma_{\text{data}}$, the final noised distribution $p(\mathbf{x};\sigma_{\max})$ is essentially indistinguishable from random noise.

Karras et al. (2022) consider a probability-flow ODE with the corresponding continuous noise schedule $\sigma(t)$ that maintains the desired distribution as $\mathbf{x}$ evolves through time given by Equation (2.19).

$$\mathrm{d}\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma(t))\mathrm{d}t \tag{2.19}$$

where the dot indicates a time derivative and $\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma(t))$ is the score function (Hyvärinen, 2005), which points towards the data at a given noise level. Infinitesimal forward or backward steps of this ODE either nudge a sample away or towards the data. Karras et al. (2022) consider training a denoiser $D_\theta(\mathbf{x};\sigma)$ on an L2 denoising objective:

$$\min_\theta \mathbb{E}_{\mathbf{x}\sim p,\sigma,\epsilon\sim\mathcal{N}(0,\sigma^2 I)} \left\| D_\theta(\mathbf{x}+\epsilon;\sigma) - \mathbf{x} \right\|_2^2 \tag{2.20}$$

and then use the connection between score-matching and denoising (Vincent, 2011) to obtain $\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma) = (D_\theta(\mathbf{x};\sigma) - \mathbf{x})/\sigma^2$. We may then apply an ODE (or SDE as a generalization of Equation (2.19)) solver to reverse the forward process.

2.3 Part II: Reinforcement Learning from Offline Data

In the final section of the background, we introduce offline reinforcement learning (Ernst et al., 2005; Levine et al., 2020) which is the key focus of the second part of the thesis. Reinforcement learning as presented in Section 2.1 fundamentally

assumes that the agent is able to interact with the environment online to collect data. However, in many real-world settings, such as robotics or clinical decision-making, online interactions can be expensive or impossible to perform due to physical or safety constraints. Furthermore, learning from scratch (or tabula-rasa) disregards any prior data that may be available on the environment, particularly when expert demonstrations are available. This is orthogonal to the previous section on model-based methods, which explored a complementary direction on efficiently making use of available data. In many cases, leveraging offline data in the learning process can significantly accelerate learning and improve robustness. In this section, we introduce the offline reinforcement learning problem, the necessary adjustments to standard online algorithms to make them work offline, and online reinforcement learning augmented with offline data.

2.3.1 Problem Setting and Challenges

In offline reinforcement learning, the policy is not deployed in the environment until test-time. Instead, the algorithm only has access to a static dataset $\mathcal{D}_{\text{env}} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$, collected by one or more behavioral policies π_b. We refer to the distribution from which $\mathcal{D}_{\text{env}}$ was sampled as the *behavioral distribution* (Yu et al., 2020b). The goal of offline RL is the same as in the online setting: to learn a policy π_θ that maximizes the expected return.

The offline reinforcement learning problem is challenging because the agent is restricted to the offline dataset $\mathcal{D}_{\text{env}}$ and cannot collect additional data. This prevents naïve application of online RL algorithms to the offline setting. On-policy algorithms as described in Section 2.1.3 which require data in-distribution with the current policy are totally infeasible due to the distribution shift. Even off-policy algorithms such as SAC (Haarnoja et al., 2018) rely on the ability to evaluate counterfactuals, i.e. over different actions for the Q-function, which may be mis-estimated offline. This issue can lead to extreme overestimation of the value function (Kumar et al., 2019) and poor performance. To illustrate this further,

consider the standard form of the Q-learning update which maximizes over the next action given an experience tuple $(s, a, r, s') \sim \mathcal{D}_{\text{env}}$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{2.21}$$

where α is the learning rate. If (s', a') is not contained in $\mathcal{D}_{\text{env}}$, then the Q-function will never be explicitly updated at this state-action pair. Thus, any extrapolation error present in the estimate of $Q(s', a')$ will be propagated throughout the learning process and state-actions with low real return can be overestimated. In the online setting, this type of overestimation is avoided by the ability to actually take the action a' in state s' and obtaining corrective feedback from the environment.

2.3.2 Solution Methods

Algorithms that can learn from offline data can be broadly categorized into two classes: model-free and model-based with a different approach to tackling extrapolation error.

Model-free. Due to the extrapolation error issues outlined in Section 2.3.1, model-free methods in offline RL typically rely on constraining the policy to lie within the support of the behavioral policy π_b. This means ensuring that $\pi_\theta(a|s)$ only has positive density where $\pi_b(a|s) > 0$. One of the first successful methods to do in the deep setting is Batch-Constrained Deep Q-Learning (BCQ, Fujimoto et al. (2019)) which uses a generative model to estimate the support of the behavioral policy.

Concretely, BCQ trains a conditional variational autoencoder (CVAE, Sohn et al. (2015)) $G_\omega(s)$ to learn the conditional behavioral distribution $\pi_b(a|s)$ and then uses this model to generate actions for the policy and Q-function update. To increase the diversity of these actions, a perturbation model $\xi_\phi(s, a, \Phi)$ outputs an adjustment to an action a in the range $[-\Phi, \Phi]$ trained to maximize the Q-function. The Q-function thus maximizes over a set of n perturbed actions $\{a_i + \xi_\phi(s, a_i, \Phi)\}_{i=1}^n$. where $a_i \sim G_\omega(s)$. The choice of n and Φ trades off between behavioral cloning

(simply copying the behavioral distribution) and full reinforcement learning. When $n = 1$ and $\Phi = 0$, BCQ is equivalent to behavioral cloning (Bain and Sammut, 1995; Bratko et al., 1995) Conversely, when $n \to \infty$ and Φ covers the full action space, BCQ is equivalent to full reinforcement learning.

Our later Chapters 4 and 7 consider a more recent minimalist approach to policy constraints, TD3+BC (Fujimoto and Gu, 2021) which avoids the need for any model of the behavioral policy. This method achieves state-of-the-art performance on a variety of offline RL benchmarks with minimal hyperparameter tuning. Concretely, TD3+BC proposes to augment the TD3 (Fujimoto et al., 2018) policy update with the following behavioral cloning term:

$$\pi = \arg\max_{\pi'} \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{env}}} \left[\lambda Q(s, \pi'(s)) - (\pi'(s) - a)^2\right] \tag{2.22}$$

where $\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i,a_i)|}$ is an adaptive normalization term computed over each batch with fixed hyperparameter α. λ scales the behavioral cloning term based on the average magnitude of the Q-values over a minibatch which is necessary as the Q-values can grow over time and dominate the loss.

Model-based. Learning a model as in Section 2.2 presents an alternative approach to avoiding extrapolation error by allowing a policy to take out-of-distribution actions in a simulated model of the environment; these methods are used in Chapters 3, 6 and 7. Whilst this provides us the opportunity to get critical feedback on out-of-distribution actions, we now instead have to contend with dynamics modeling errors from a fixed dataset. One of the first successful offline model-based reinforcement learning algorithms was Model-based Offline Policy Optimization (MOPO, Yu et al. (2020b)) which extends MBPO to the offline setting. MOPO defines the notion of a pessimistic MDP (P-MDP) which provides a theoretical lower bound on the expected true return.

Given a learned dynamics model $\widehat{P}$ similar to Section 2.2.1, we define the model MDP $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma)$. We then separately denote the return under the true

MDP, $J_M(\pi)$, and the return under the model MDP, $J_{\widehat{M}}(\pi)$. Yu et al. (2020b) show that when the estimated reward matches the true reward, i.e. $\widehat{R} = R$

$$J_M(\pi) \geq \mathbb{E}_{\pi,\widehat{P}}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) - \gamma |G^{\pi}_{\widehat{M}}(s_t, a_t)|\right)\right] \tag{2.23}$$

where

$$G^{\pi}_{\widehat{M}}(s, a) = \mathbb{E}_{s' \sim \widehat{P}(\cdot|s,a)}\left[V^{\pi}_{M}(s')\right] - \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V^{\pi}_{M}(s')\right] \tag{2.24}$$

is a measure of discrepancy between the true and model dynamics as measured by the value function. Since we do not have access to the true dynamics, various heuristics for $|G^{\pi}_{\widehat{M}}(s, a)|$ are instead proposed. In practice, MOPO optimizes the following objective

$$J_{\text{MOPO}}(\pi) = \mathbb{E}_{\pi,\widehat{P}}\left[\sum_{t=0}^{\infty} \gamma^t \left(\widehat{R}(s_t, a_t) - \lambda \cdot u(s_t, a_t)\right)\right] \tag{2.25}$$

where $u(s_t, a_t)$ is a measure of uncertainty in the learned dynamics model, e.g. the predictive variance of the model, and λ is a scaling factor. Concurrent to MOPO, MOReL (Kidambi et al., 2020) introduced a different P-MDP by augmenting a standard MDP with a negative valued absorbing state that is transitioned to when the total variation distance between true and learned dynamics is exceeded.

2.3.3 Online Reinforcement Learning with Prior Data

Finally, we introduce the related setting of online reinforcement learning augmented with prior data, which is the focus of Chapter 5. In many cases, we may not be restricted to completely learning offline, but instead have a small budget of online steps available to us in order to fine-tune an offline policy. When we have no assumption on the quality of the offline data, there is a rich body of work that fine-tunes a learned offline policy with a few online samples (Kostrikov et al., 2022; Lee et al., 2022; Nair et al., 2020). However, when we know that the offline data is composed of high-quality expert demonstrations, we can instead leverage Kullback-Leibler (KL) regularized methods (Galashov et al., 2019; Rawlik et al., 2012; Schulman et al., 2017a; Todorov, 2007).

Given a set of expert demonstrations *without reward*, $\mathcal{D}_0 = \{(s_n, a_n)\}_{n=1}^N$, we first use supervised behavioral cloning to learn a base policy $\pi_0 : \mathcal{S} \to \mathcal{A}$. Since expert demonstrations are costly to obtain and often only available in small number, behavioral cloning alone is typically insufficient for agents to learn good policies in complex environments and has to be complemented by a method that enables the learner to build on the cloned behavior by interacting with the environment. KL-regularized reinforcement learning modifies the standard reinforcement learning objective by augmenting the return with a negative reverse KL divergence term from the learned policy π to the reference policy π_0, given a temperature parameter α. The resulting regularized return is then given by

$$\tilde{J}(\pi) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) - \alpha \mathrm{KL}(\pi(\cdot|s_t) \parallel \pi_0(\cdot|s_t))\right)\right] \tag{2.26}$$

Crucially, the KL-divergence also allows us to leverage any uncertainty in π_0 to guide exploration. The agent is discouraged to explore areas of the state space $\mathcal{S}$ where the variance of $\pi_0(\cdot|s)$ is low (i.e., more certain) and encouraged to explore areas of the state space where the variance of $\pi_0(\cdot|s)$ is high.

Part I

Synthetic Environments and Data for Reinforcement Learning

3

Augmented World Models Facilitate Zero-Shot Dynamics Generalization From a Single Offline Environment

We begin the research chapters of the thesis by investigating how synthetic data can help reinforcement learning agents better generalize to new tasks. We specifically study the setting where we have offline data (Ernst et al., 2005; Levine et al., 2020) on *only a single environment*, but where the test-time dynamics may differ (Rakelly et al., 2019b; Zintgraf et al., 2021a,b). This is a particularly realistic setting as we would expect a human operator that could drive a car in dry weather would also be able to drive it in the rain where the vehicle responds differently. Generalization from a single task is also considerably harder than settings considered by prior model-based generalization works, which assume access to a distribution of training tasks (Kirchmeyer et al., 2022; Lee et al., 2020). To motivate our approach, we start by considering variants of the standard MuJoCo (Todorov et al., 2012) locomotion tasks where the mass and damping of the robots may vary. We show that baseline MOPO (introduced in Section 2.3.2) policies using single-task data from the standard D4RL (Fu et al., 2020) benchmark fail to generalize to changes in test time dynamics, with performance dropping up to 90%.

We propose a solution, *Augmented World Models* (AugWM), which is based on perturbing a learned world model in order to capture potential changes in the physical properties of a robot. We consider several families of augmentations, with the best being *Dynamics Amplitude Scaling* (DAS). DAS scales the *change in the state* which we denote with $\delta_t = s_{t+1} - s_t$ by mapping $s_{t+1} \mapsto s_t + z \odot \delta_t$ for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$ over a pre-defined range $[a, b]$. Intuitively, DAS dampens or amplifies the relative effects of actions, which makes it particularly suitable for modified dynamics in proprioceptive environments. This represents a novel approach to defining a distribution of tasks in the meta-learning sense, where the context is a noise vector z. We may then train a contextual policy by concatenating the noise vector z onto the state, which remarkably already aids generalization.

While many realizations of the task z may correspond to invalid robot configurations, our goal is simply to better cover the test distribution. We confirm that this indeed happens—in a modified dynamics MuJoCo task where we have oracle access to the ground truth dynamics, passing in the exact change in dynamics to our contextual policy allows the agent to recover the original performance. Even without the oracle, the unique form of the augmentation allows us to estimate the true augmentation at test time. A simple linear model trained online at test time comparing the true change in state to the model prediction for the original environment produces a useful signal for predicting the correct augmentation vector z.

We rigorously evaluate our approach on over 100 different changed dynamics settings, and show that this simple approach can significantly improve zero-shot generalization. Notably, we show that we can also handle more complex modified dynamics, such as crippled legs and modified limb sizes. Finally, we show that AugWM is flexible enough to allow simulated robots to adapt to dynamics changes *mid-episode*; for example, when a robot suffers a joint failure at run-time.

Philip J. Ball*, **Cong Lu***, Jack Parker-Holder, and Stephen J. Roberts. Augmented World Models Facilitate Zero-Shot Dynamics Generalization From a Single Offline Environment. **In ICML, 2021.**

Augmented World Models Facilitate Zero-Shot Dynamics Generalization From a Single Offline Environment

Philip J. Ball [*1] **Cong Lu** [*1] **Jack Parker-Holder** [1] **Stephen Roberts** [1]

Abstract

Reinforcement learning from large-scale offline datasets provides us with the ability to learn policies without potentially unsafe or impractical exploration. Significant progress has been made in the past few years in dealing with the challenge of correcting for differing behavior between the data collection and learned policies. However, little attention has been paid to potentially changing dynamics when transferring a policy to the online setting, where performance can be up to 90% reduced for existing methods. In this paper we address this problem with *Augmented World Models* (AugWM). We augment a learned dynamics model with simple transformations that seek to capture potential changes in physical properties of the robot, leading to more robust policies. We not only train our policy in this new setting, but also provide it with the sampled augmentation as a context, allowing it to adapt to changes in the environment. At test time we *learn the context* in a self-supervised fashion by approximating the augmentation which corresponds to the new environment. We rigorously evaluate our approach on over 100 different changed dynamics settings, and show that this simple approach can significantly improve the zero-shot generalization of a recent state-of-the-art baseline, often achieving successful policies where the baseline fails.

1. Introduction

Offline reinforcement learning (RL) describes the problem setting where RL agents learn policies solely from previously collected experience without further interaction with the environment Fujimoto et al. (2019); Levine et al. (2020). This could have tremendous implications for real-world problems Dulac-Arnold et al. (2019), with the potential to leverage rich datasets of past experience where exploration is either not feasible (e.g. a Mars Rover) or unsafe (e.g. in medical settings). As such, interest in offline RL has surged in recent times.

This work focuses on model-based offline RL, which has achieved state-of-the-art performance through the use of uncertainty penalized updates Yu et al. (2020); Kidambi et al. (2020). However, existing work only addresses the issue of transferring from different behavior policies in the *same environment*, ignoring any possibility of distribution shift. Consider the case where it is expensive to collect data, and we have access to a single dataset from a robot. Using existing methods, we would be unable to make any changes that impact the dynamics, such as using a newer model of the robot or deploying it in a different room.

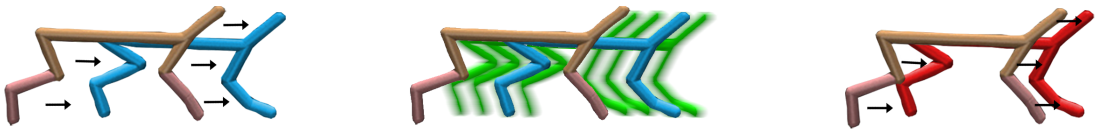
A related setting is the Sim2Real problem, which considers transferring an agent from a simulated environment to the real world. A popular recent approach is domain randomization Tobin et al. (2017); James et al. (2017), the process of randomizing non-essential regions of the observation space to make agents robust to 'observational overfitting' Song et al. (2020). Indeed, methods seeking to generalize to novel dynamics have also shown promise Peng et al. (2018), by randomizing physical properties such as the mass of the agent. A significant limitation of these approaches is the requirement for a simulator, which may not be available.

In this work we take inspiration from Sim2Real to generalize solely from an offline dataset, in a learned simulator or *World Model* (WM). We therefore describe our problem setting as follows: an agent must learn to generalize to unseen test-time dynamics whilst having access to offline data from *only a single environment*; we call this "dynamics generalization from a single offline environment".

In this paper we concentrate on the zero-shot performance of our agents to unseen dynamics, as it may not be practical (nor safe) to perform multiple rollouts at test time. To tackle this problem, we propose a novel form of data augmentation: rather than augment observations, we focus on *augmenting the dynamics*. We first learn a world model of the environment, and then augment the transition function at

[*]Equal contribution [1]University of Oxford, United Kingdom. Correspondence to: Philip Ball <ball@robots.ox.ac.uk>, Cong Lu <cong.lu@stats.ox.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.



(a) World Model: $\widehat{P}$ (b) Augmented World Model: $\widehat{P}_z$ (c) Test: $P^\star$

Figure 1. An illustration of our approach, each figure shows a transition $P(s, a) \to s'$. In a) we show the World Model dynamics $\widehat{P}$, trained from $\mathcal{D}_{\text{env}}$, a single offline dataset. In b) we show the Augmented World Model, blue represents $\widehat{P}$, while each green agent illustrates an instantiation of augmented dynamics, which is sampled at each timestep: $\widehat{P}_z, z \sim \mathcal{Z}$. The goal is to approximate c) where we show an unseen test environment, with transition dynamics $P^\star$.

policy training time, making the agent train under different imagined dynamics. In addition, our agent is given access to the augmentation itself as part of the observations, allowing it to consider the context of modified dynamics.

At test time we propose a simple, yet surprisingly effective, self-supervised approach to learning an agent's augmentation context. We learn a linear dynamics model which is then used to approximate the dynamics augmentation induced by the modified environment. This context is then given to the agent, allowing it to adapt on the fly to the new dynamics within a single episode (i.e. zero-shot). We show that our approach is capable of training agents that can vastly outperform existing Offline RL methods on the "dynamics generalization from a single offline environment" problem. We also note that this approach does *not* require access to environment rewards at test time. This facilitates application to Sim2Real problems whereby test time rewards may not be available.

Our contributions are twofold: 1) As far as we are aware, we are the first to propose **dynamics augmentation for model based RL**, allowing us to generalize to changing dynamics despite only training on a single setting. We do this without access to any environment parameters or prior knowledge. 2) We propose a simple **self-supervised context adaptation** reward-free algorithm, which allows our policy to use information from interactions in the environment to vary its behavior in a single episode, increasing zero-shot performance. We believe both of these approaches are not only novel, but offer significant improvement v.s. state-of-the-art methods, improving generalization and providing a promising approach for using offline RL in the real world.

2. Related Work

In this work we focus on Model Based RL (MBRL). A key challenge in MBRL is that an inaccurate model can be exploited by the policy, leading to behaviors that fail to transfer to the real environment. As such, a swathe of recent works have made use of model ensembles to improve robustness Kurutach et al. (2018); Chua et al. (2018); Clavera et al. (2018); Janner et al. (2019); Ball et al. (2020). With increased accuracy, MBRL has recently been shown to be competitive with model free methods in continuous control Ha and Schmidhuber (2018); Chua et al. (2018); Janner et al. (2019) and games Schrittwieser et al. (2019); Kaiser et al. (2020). We make use of an ensemble of probabilistic dynamics models, first introduced in Lakshminarayanan et al. (2017) and subsequently used in Chua et al. (2018).

In this paper we focus on Model-Based *offline* RL, where MOPO Yu et al. (2020) and MOReL Kidambi et al. (2020) have recently demonstrated the effectiveness of learned dynamics models, using model uncertainty to constrain policy optimization. We build upon this approach for zero-shot dynamics generalization from offline data. There have also been successes in off policy methods for offline RL Wu et al. (2019); Fujimoto et al. (2019); Kumar et al. (2020); Rudner et al. (2021) and context based approaches Ajay et al. (2021), although these works only consider tasks within the support of the offline dataset. Finally, MBOP Argenson and Dulac-Arnold (2021) addresses the problem of goal-conditioned zero-shot transfer from offline datasets. However, their goal-conditioning relies on unchanged dynamics in the test environment.

In online RL, recent work has achieved strong dynamics generalization with a learned model Seo et al. (2020). However, this required training under varied dynamics, assigning different experiences to models. In addition, this work used MPC whereas we train a policy inside the model, which is significantly faster at deployment time. Also related are Clavera et al. (2019); Nagabandi et al. (2019), where the model is trained to quickly to adapt to new dynamics $P(s'|s, a)$, however both these works place more emphasis on model-adaption rather than zero-shot policy performance. Furthermore, access to an underlying task distribution is required, something we do not have in our offline setting. Also similar to our work is the recently proposed Policy Adaptation during Deployment (PAD, Hansen et al. (2021)) approach. Our approach differs in that we learn a context, whereas PAD uses an auxiliary objective to adapt its features. In addition, PAD considers the *online model free* setting, while our method is *offline and model based*.

Sim2Real is the setting where an agent trained in a simulator must transfer to the real world. A common approach to

solve this problem is through domain randomization Tobin et al. (2017); James et al. (2017), whereby parameters in the simulator are varied during training. This has shown to be effective for dynamics generalization Andrychowicz et al. (2020); Antonova et al. (2017); Peng et al. (2018); Yu et al. (2017); Zhou et al. (2019); OpenAI et al. (2019), but requires access to a simulator which we do not have. Another form of domain randomization, data augmentation, has proved to be effective for training RL policies Laskin et al. (2020a;b); Kostrikov et al. (2021); Raileanu et al. (2020), resulting in improved efficiency and generalization. So far, these works have focused on online model free methods, and used data augmentation on the *state* space, reducing *observational* overfitting Song et al. (2020). In contrast, we focus on offline MBRL and instead augment the *dynamics*.

We also note clear links to contextual MDPs Hallak et al. (2015); Modi et al. (2018) and hidden parameter MDPs (HiP-MDP) Doshi-Velez and Konidaris (2016); Killian et al. (2017); Zhang et al. (2021) settings, whereby our self-supervised dynamics embedding can be considered as a context/hidden parameter. However, in these settings the embedding is chosen at the beginning of each episode and is fixed throughout, whereas our embedding varies per timestep.

We are not the first to propose data augmentation in the MBRL setting, Pitis et al. (2020) proposed Counterfactual data augmentation for improving performance in the context of locally factored tasks. Approaches to ensuring adversarial robustness can include data augmentations that assist with out-of-domain generalization, as opposed to observational overfitting. In Volpi et al. (2018) this is done without a simulator and from a single source of data, however they only work on supervised learning problems and require an adversary to be learned, adding computational complexity. Finally, Wellmer and Kwok (2021) concurrently explore the idea of augmenting world model dynamics for improved test-time transferability, however they focus on in-domain generalization, and do not infer context at test time.

3. Preliminaries

We consider a Markov Decision Process (MDP), defined as a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space respectively, $P(s'|s,a)$ the transition dynamics, $R(s,a)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in (0,1)$ the discount factor. The goal in RL is to optimize a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right]$. The value function $V^\pi(s) := \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s\right]$ gives the expected discounted return under π when starting from state s. In *offline RL*, the policy is not deployed in the environment until test time. Instead, the algorithm only has access to a static dataset $\mathcal{D}_{env} = \{(s,a,r,s')\}$, collected by one or more behavioral policies π_b. We borrow notation from Yu et al. (2020) and refer to the distribution from which $\mathcal{D}_{env}$ was sampled as the *behavioral distribution*.

When training a model, we follow MBPO Janner et al. (2019) and MOPO Yu et al. (2020) and train an ensemble of N probabilistic dynamics models Nix and Weigend (1994). Each model learns to predict both next state s' and reward r from a state-action pair, using $\mathcal{D}_{env}$ in a supervised fashion. Furthermore, each model outputs a Gaussian $\widehat{P}_i(s_{t+1}, r_t|s_t,a_t) = \mathcal{N}(\mu(s_t,a_t), \Sigma(s_t,a_t))$. The resulting model $\widehat{P}$ defines a *model MDP* $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \widehat{R}, \rho_0, \gamma)$, where $\widehat{R}$ refers to the learned reward model.

To train the policy, we use k step rollouts inside $\widehat{M}$, adding experience to a replay buffer $\mathcal{D}_{\widehat{\text{env}}}$ to learn an action-value function and a policy, using Soft Actor Critic (SAC Haarnoja et al. (2018)). SAC alternates between a soft policy evaluation step, which estimates $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)))|s_0=s, a_0=a]$ using Bellman backups (where α is a temperature parameter for policy entropy $\mathcal{H}$), and a policy improvement step which learns a policy by minimizing the expected KL divergence $J_\pi(\phi,\mathcal{D}) = \mathbb{E}_{s_t\sim\mathcal{D}_{\widehat{\text{env}}}}[D_{KL}(\pi\|\exp\{\frac{1}{\alpha}\{Q^\pi - V^\pi\}\})]$. Note that the SAC algorithm is unchanged from the model-free setting, aside from the environment being a learned model, and the rollout horizon k being truncated. Perhaps surprisingly, this approach alone produces strong results in the offline setting. However, MOPO Yu et al. (2020) and MOReL Kidambi et al. (2020) show improvement in performance by penalizing rewards in regions of the state space where the ensemble of probabilistic models is less certain. This implicitly reduces reliance on samples which deviate beyond the support of $\mathcal{D}_{env}$.

While MOPO and MOReL have addressed the issue of training a policy in $\mathcal{D}_{env}$, and transferring to the true environment M, they only consider where the data in $\mathcal{D}_{env}$ is actually drawn from P. However, sometimes this may not be sufficient for deployment. For example, a robot could fail to walk when learning from data that was collected by a different version of the robot (with different mass), or if the same robot collected data but in a different room to deployment (with varied friction). It is this setting, where dynamics may vary at test time, that is the focus of our work. To learn successfully, we propose a novel approach to training robust context-dependent policies.

4. Augmented World Models with Self Supervised Policy Adaptation

In this section we introduce our algorithm: Augmented World Models (AugWM). We first discuss our training procedure (Fig. 2) before moving onto our self-supervised approach to online context learning (Fig. 3).

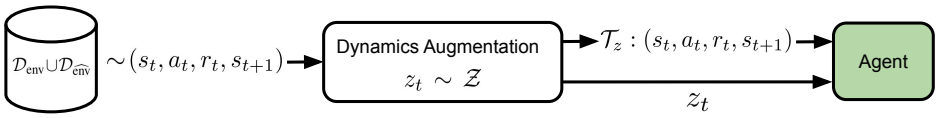


Figure 2. Training policies in Augmented World Models. For each state-action pair sampled from the buffer, a new augmentation $z_t \sim \mathcal{Z}$ is sampled to produce an augmentation operator $\mathcal{T}_z$, which is applied to the transition. The policy is then trained with the new tuple of data, with the context concatenated to the state.

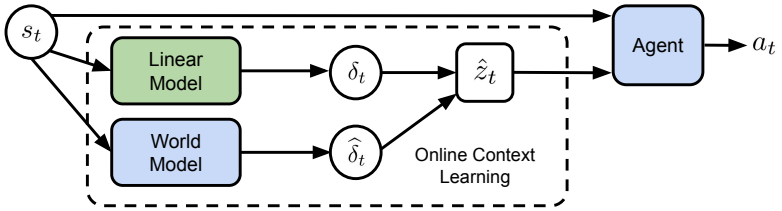


Figure 3. Self-supervised policy adaptation via learned context. At each timestep, the state s_t is fed into a linear model (trained online at each timestep) to predict the change in state δ_t, and also passed to the fixed World Model (trained on the offline data) to predict the change in state under $\widehat{P}$, $\widehat{\delta}_t$. The approximate context is then $z_t = {}^{\delta_t}/_{\hat{\delta}_t}$, which is concatenated with s_t and passed to the agent to produce an action.

4.1. Augmented World Models

Rather than seeking to transfer our policy from $\widehat{M}$ to M, we instead wish to transfer from $\widehat{M}$ to $M^\star$, where $M^\star = (\mathcal{S}, \mathcal{A}, P^\star, R^\star, \rho_0, \gamma)$ is an unseen environment with *different dynamics*. Thus, our emphasis shifts to producing experience inside the world model such that our agent is able to generalize to unseen, out of distribution dynamics. We approach this problem by training our policy in an *Augmented World Model*. Formally, we denote an augmentation as $z_t \sim \mathcal{Z}, z \in \mathbb{R}^{|\mathcal{S}|}$, which is sampled at each timestep to produce an augmentation operator $\mathcal{T}_z$. $\mathcal{T}_z$ is applied to (s, a, r, s') tuples from the dataset $\mathcal{D}$, and when used with tuples sampled from $\mathcal{D}_{\widehat{\text{env}}}$ indirectly induces an augmented distribution $\widehat{P}_z$. In principle, we wish to produce augmentations such that the true modified environment dynamics $P^\star$ lies in the support of the distribution of augmented dynamics, i.e.

$$\inf_z D(\widehat{P}_z(s,a) \| P^\star(s,a)) \leq \epsilon \tag{1}$$

for all s, a, some small $\epsilon > 0$, and suitable distance/divergence metric D. We consider several augmentations, beginning with existing works before moving to new approaches which specifically target the problem of dynamics generalization. We begin with **Random Amplitude Scaling** as in Laskin et al. (2020a), which we refer to as RAD. RAD scales both s_t and s_{t+1} as follows:

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (z \odot s_t, a_t, r_t, z \odot s_{t+1}) \tag{2}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. Given that our focus is on changing dynamics (vs. observational overfitting), we also propose to scale only the next state, i.e., **Random Amplitude Nextstate Scaling** (RANS):

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (s_t, a_t, r_t, z \odot s_{t+1}) \tag{3}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. Note that while RANS is more focused on augmenting dynamics than RAS, it still suffers from a dependence on the magnitude of s_{t+1}. As such, we further propose a more targeted augmentation, which we call **Dynamics Amplitude Sampling** (DAS). Rather than directly scale the state, DAS scales the *change in the state* which we denote with $\delta_t = s_{t+1} - s_t$, as follows:

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (s_t, a_t, r_t, s_t + z \odot \delta_t) \tag{4}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. The full training procedure is shown in Algorithm 1.

Algorithm 1 Augmented World Models: Training

Input: Offline data $\mathcal{D}_{\text{env}}$, Penalty λ, horizon h, batchsize B, augmentation $\mathcal{Z}$.
Initialize: Ensemble of N dynamics models $\widehat{P}$, policy π. Replay buffer $\mathcal{D}_{\widehat{\text{env}}}$
1. Train $\widehat{P}$ in a supervised fashion using $\mathcal{D}_{\text{env}}$.
for *epoch* $= 1, 2, \ldots$ **do**
  (i) Sample initial states: $\{s_1^1, \ldots, s_1^B\} \sim \mathcal{D}_{\text{env}}$
  (ii) Rollout policy (in parallel), using λ-penalized reward, storing all data in $\mathcal{D}_{\widehat{\text{env}}}$
  (iii) Train policy using $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\widehat{\text{env}}}$. For each (s, a, r, s'), sample $z \sim \mathcal{Z}$ and apply $\mathcal{T}_z$.
end
Return: Policy π

One crucial addition to our method is the use of context. Concretely, when we are optimizing the policy using a batch of data, we concatenate the next state with the augmentation

vector z. This allows our policy to be informed of the specific augmentation that was applied to the environment and thus behave accordingly. However, at test time we do not know z, so what can we use? Next we propose a solution to this problem, learning the context on the fly.

4.2. Self-Supervised Policy Selection

In the meta-learning literature there have been many recent successes making use of a *learned context* to adapt a policy at test time to a new environment Rakelly et al. (2019); Zintgraf et al. (2020; 2021), typically using a blackbox model with a latent state. Crucially, these approaches require several episodes to adapt at test time, making them unfeasible in our zero-shot setting. What makes our setting unique is we explicitly know what the context represents: a linear transformation of s_t, or δ_t. Using this insight, we are able to learn an effective context on the fly at test time. Concretely, we observe that from a state s_t drawn from $M^\star$, we can sample an action $a_t \sim \pi$ and then compute an approximate $\widehat{s}_{t+1}$ using our model $\widehat{P}$. With $\widehat{s}_{t+1}$, we have a sample estimate of the *state change under* $\widehat{P}$, i.e. $\widehat{\delta}_t = \widehat{s}_{t+1} - s_t$. We can make this approximation of the next state without interacting with the environment, but once we do take the action a_t in the environment, we then receive the *true next state* s_{t+1} and can store the true difference $\delta_t = s_{t+1} - s_t$. Using the DAS augmentation, we can approximate z as ${}^{\delta}/_{\hat{\delta}}$.

Algorithm 2 Augmented World Models: Testing

Input: Initial state s_1, horizon H, policy π, world model $\widehat{P}$, initial context $\hat{z} = \mathbf{1}^{|\mathcal{S}|}$
Initialize: Linear model f_ψ, dataset $\mathcal{D} = \emptyset$, return $R_0 = 0$.
for *step* $= 1, 2, \dots H$ **do**
  Select action: $a_t \sim \pi(s_t, \widehat{z}_t)$
  Take action: $s_{t+1} \sim P^\star(s_t, a_t)$
  Update return: $R_{t+1} = R_t + R^\star(s_t, a_t, s_{t+1})$
  Update Dataset: $\mathcal{D} \cup (s_t, \delta_t)$.
  Update Linear model by minimizing $\mathcal{L}_{\text{MSE}}(\psi, \mathcal{D})$.
  Predict new context using $\widehat{z}_t = {}^{\delta}/_{\hat{\delta}}$.
end
Return: R_T

This however is retrospective; we can only approximate z having already seen the next state, by which time our agent has already acted. Furthermore, we believe under changed dynamics the true z likely depends on s, thus we cannot use a previous z for future timesteps. Therefore, we learn a forward dynamics model using the data collected *during* the test rollout. After h timesteps in the environment, we have the following dataset: $\mathcal{D} = \{(s_1, s_2), \dots, (s_{h-1}, s_h)\}$. This allows us to learn a simple dynamics model $f_\psi : (s_t) \mapsto \delta_t = s_{t+1} - s_t$, by minimizing the mean squared error $\mathcal{L}_{\text{MSE}}(\psi, \mathcal{D})$. Notably, since we never actually plan with this model, it does not need to be as accurate as a typical dynamics model in MBRL. Instead, it is crucial that the model learns quickly enough such that we can use it in a zero-shot evaluation. Thus, we choose to use a simple linear model for f. To show the effectiveness of this, in Fig. 4 we show the mean R-squared of linear models learned on the fly during evaluation rollouts.

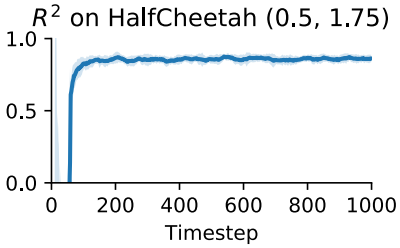


Figure 4. Mean R^2 of the linear model across 20 rollouts.

We observe that in less than 100 timesteps the linear model achieves high accuracy on the test data. Subsequently, equipped with f_ψ, we can approximate δ_t, and predict the augmentation as $\widehat{z}_t = {}^{\delta}/_{\hat{\delta}}$. We then provide the agent with $\widehat{z}_t$ to compute action a_t. The full procedure is shown in Algorithm 2.

5. Experiments

In our experiments, we aim to investigate the effectiveness of our approach for zero-shot dynamics generalization from a single offline dataset. To assess this, we will answer a series of questions, beginning with a question on the necessity of our method:

Do we really need to develop methods specifically for dynamics generalization?

To answer this, we train MOPO using offline data from a single environment, and test it under changed dynamics. We consider the HalfCheetah environment from the OpenAI Gym Brockman et al. (2016), using offline data from D4RL Fu et al. (2021). We train a MOPO agent using the mixed dataset, using our own implementation of the algorithm (but using the same hyperparameters as the original authors). To test the trained policy, we vary both the mass of the agent and damping coefficient by a multiplicative factor.[1] In this work we consider a grid of the following values for HalfCheetah: $\{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$ and $\{0.5, 0.75, 1.0, 1.25, 1.5\}$ for Walker2d, representing a significant out-of-distribution shift.

The results (Fig. 5), show that MOPO performance is clearly impacted by changing dynamics. We see in the central cell, that performance for our version of MOPO matches the author results Yu et al. (2020), and in some cases we even see small gains (e.g. mass $= 0.75$, damping $= 1.0$). However,

[1]The standard environment (both in Gym and D4RL) corresponds to both these values being set to 1.0.

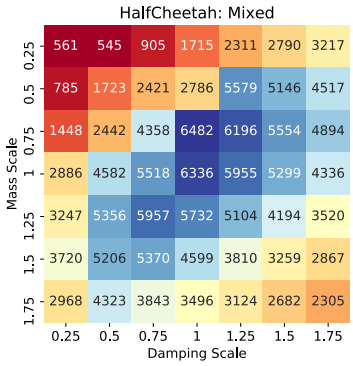


Figure 5. Mean performance for 5 seeds for MOPO on HalfCheetah environments with varying dynamics. Note the central cell (1,1) corresponds to the in-sample data.

on the top left we see dramatically weaker performance, often below 1k, indicating the robot is failing to achieve locomotion. Before evaluating AugWM, we first test whether training with the "correct" augmentation improves generalization performance. In short, we ask:

Is augmenting dynamics even worthwhile?

To answer this, we train SAC for 1×10^5 steps and save the states visited in the 'true' environment. We then use these starting states to train a policy using an offline MBPO[2] approach with AugWM. However, instead of sampling $z_t \sim \mathcal{Z}$, we provide the actual $z = {}^{\delta}/_{\delta}$ as we have access to the 'true' and 'modified' environments; we refer to this as an *oracle* version of our method, and is designed to assess the viability of our approach. Note that we *do not* augment the 'true' environment rewards. We consider two baselines: a) offline MBPO in the 'true' environment; b) online SAC in the 'modified' environment. We train MBPO until convergence, and SAC for 1×10^5 steps.

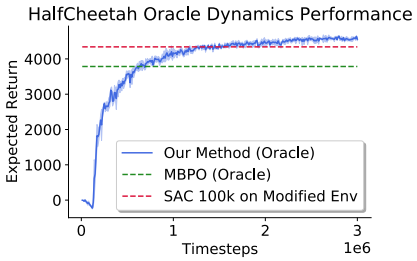


Figure 6. Mean performance for 3 seeds for MBPO with a dynamics oracle on HalfCheetah with $1.5\times$ mass and damping.

As shown Fig. 6, when provided with the true z, AugWM outperforms both baselines. The SAC result is surprising: the baseline agent was trained directly on the 'modified' environment for the same number of steps as the policy that generated our oracle starting states. One explanation is the greater exploration induced by the 'easier' dynamics of the 'true' environment. This validates our approach; if we augment the dynamics $\widehat{P}$ from a model correctly, we can generalize to unseen dynamics. In other words, neither the starting states nor rewards need to be from the 'modified' environment. With this in mind, our next question is a simple one:

Which augmentation strategy is most effective?

To test this, we train as in Algorithm 1, *without* context, to isolate the effectiveness of the training process. We use the HalfCheetah Mixed dataset and train a MOPO agent, augmenting either both s and s' (RAD), just s' (RANS) or just δ (DAS).

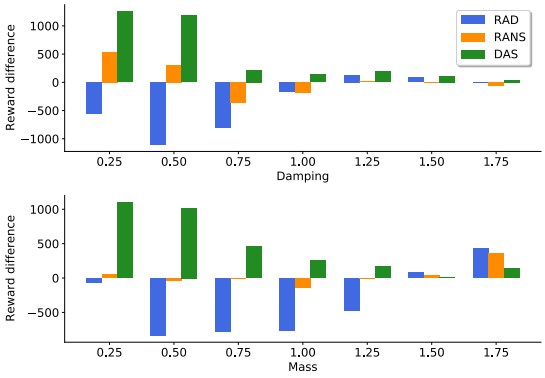


Figure 7. Average performance gains of different World Model augmentations over base MOPO for different levels of damping and mass in the HalfCheetah test environment (5 seeds).

The results are shown in Fig. 7. As we see, the RAD augmentation fails to improve dynamics generalization, actually leading to worse performance overall. RANS does improve performance on unseen dynamics, as we are influencing the *dynamics*, not just the observation. However, DAS clearly provides the strongest performance. As a result, we use DAS for AugWM. Our final algorithm design question is as follows:

Does training with context improve performance?

To answer this question, we return to the HalfCheetah Mixed setting from Fig. 7, taking the policy trained with DAS. We now train two additional agents: 1) Default Context: at *train* time the agent is provided with the DAS augmentation z as context, at *test* time it is provided with a vector of ones, $\mathbf{1}^{|\mathcal{S}|}$; 2) Learned Context: trained as in 1), but context is learned online using Algorithm 2.

The results are shown in Fig. 8. We observe that training with context (orange) improves performance on average, while adapting the context on the fly (green) leads to further gains (+80 on average). These methods combine to produce our AugWM algorithm. We are now ready for the final question:

[2]Since we have access to the true environment, there is no need for the MOPO penalty.

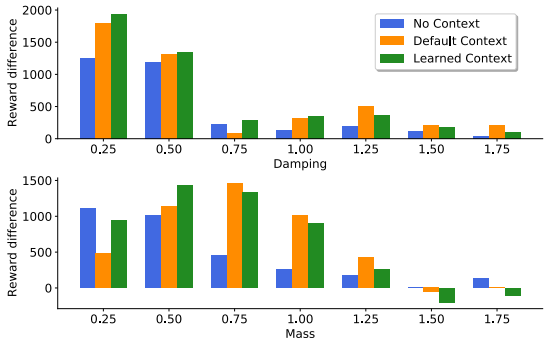


Figure 8. Average performance gains of adding contexts over base MOPO for different levels of damping and mass in the HalfCheetah test environment (5 seeds).

Can Augmented World Models improve zero-shot generalization?

To answer this question we perform a rigorous analysis, using multiple benchmarks from the D4RL dataset Fu et al. (2021). Namely, we consider the Random, Medium, Mixed and Medium-Expert datasets for both Walker2d and HalfCheetah. In each setting, we compare AugWM against base MOPO on zero-shot performance, training entirely on the data provided, but not seeing the test environment until evaluation. The results are shown as a change v.s. MOPO, averaged over one dimension in Fig. 11, and as a total return number averaged over both dimensions in Table 1. For additional implementation details (e.g., hyperparameters) see Appendix B. AugWM provides *statistically significant* improvements in zero-shot performance v.s. MOPO in many cases, achieving successful policies where MOPO fails.

Table 1. Each entry for HalfCheetah is the mean of 49 different dynamics, while for Walker2d it is over 25 dynamics. Results are mean $\pm$1std. $\star$ indicates $p < 0.05$ for Welch's t-test for gain over MOPO (5 seeds).

Dataset Type	Environment	MOPO	AugWM (Ours)
Random	HalfCheetah	2303 ± 112	2818 ± 197 $\star$
Random	Walker2d	569 ± 103	706 ± 139
Mixed	HalfCheetah	3447 ± 218	3948 ± 122 $\star$
Mixed	Walker2d	946 ± 95	1317 ± 206 $\star$
Medium	HalfCheetah	2954 ± 89	2967 ± 106
Medium	Walker2d	1477 ± 337	1614 ± 440
Med-Expert	HalfCheetah	1590 ± 766	2885 ± 432 $\star$
Med-Expert	Walker2d	1062 ± 334	2521 ± 316 $\star$

By now we have provided significant evidence that AugWM can significantly improve performance for HalfCheetah and Walker2d with varied mass and damping. However, this is only a small subset of possible dynamics changes. We next consider several significantly harder settings. We test increased dimensionality, using the Ant environment from MOPO Yu et al. (2020), and consider additional types of dynamics changes (e.g., Ant with crippled legs, HalfCheetah with changed limb sizes from Henderson et al. (2017)). We illustrate the impact of the crippled leg Ant environment on baseline agent performance, and the improvement provided by AugWM, in Fig. 12. We show the mean results over each of these factors of variation in Table 2, where again AugWM provides a non-trivial improvement over a strong baseline. For more details, see Appendix B.

Table 2. Mean performance for MOPO, AugWM with the default context, and AugWM with learned context (LM). Entries are mean zero-shot reward for all dynamics. Bold = highest (5 seeds).

Setting	MOPO	AugWM (Default)	AugWM (LM)
Ant: Mass/Damp	1634	1715	**1804**
Ant: One Crippled Leg	1370	1572	**1680**
Ant: Two Crippled Legs	700	697	**795**
HalfCheetah: Big	4891	**5194**	4968
HalfCheetah: Small	5151	**5488**	5263

Finally, we note that dynamics may change *during* an episode; consider a robot that suffers a motor fault, reducing the power delivered to its joints. Evidently the underlying dynamics have been altered, and being robust to such changes when only training from a single dataset of offline experience is challenging. To illustrate this, we perform a 1500 step rollout in the HalfCheetah environment, starting with offline dynamics (mass/damping = 1), before changing to mass = 0.75, damping = 0.5 after 500 steps; performance is shown in Fig. 9. Observe that after 500 steps, MOPO performance is dramatically reduced. This is because the agent continues to apply the same force and thus falls forward with lighter mass. For our AugWM agent, performance initially drops, then when the new context is learned we achieve *higher* performance than before, making use of the lighter torso.

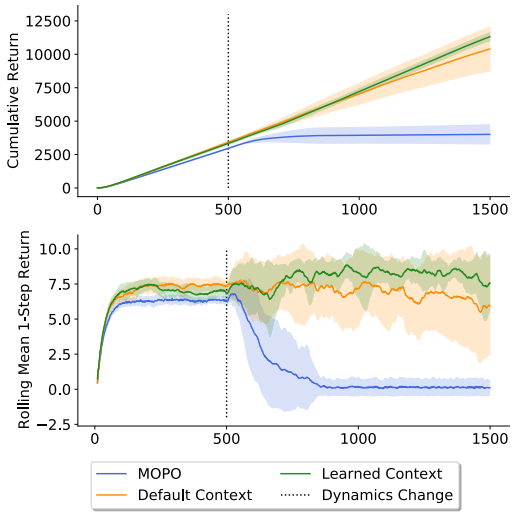


Figure 9. Performance under changing dynamics. Top = cumulative returns, Bottom = rolling average single step reward. Both averaged over twenty seeds, shaded area shows $\pm$1std.

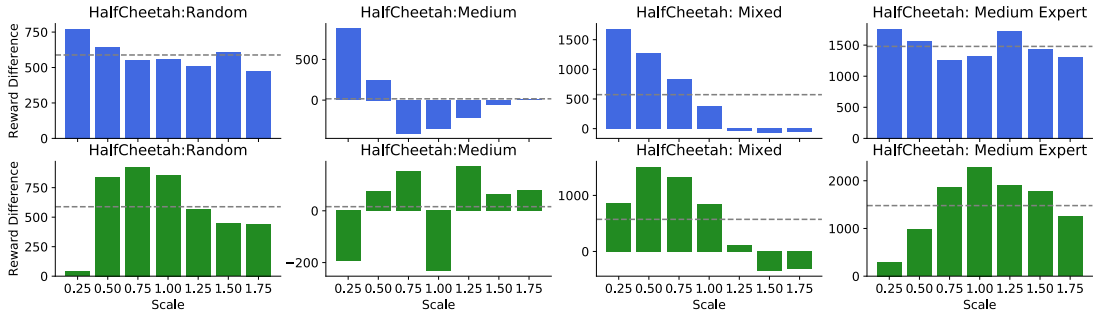


Figure 10. Mean improvement for Augmented World Models over MOPO for the HalfCheetah environment, averaged over five seeds. Top row (blue) = damping scale, bottom row (green) = mass scale. Dotted line is the mean, the same value for both damping and mass.

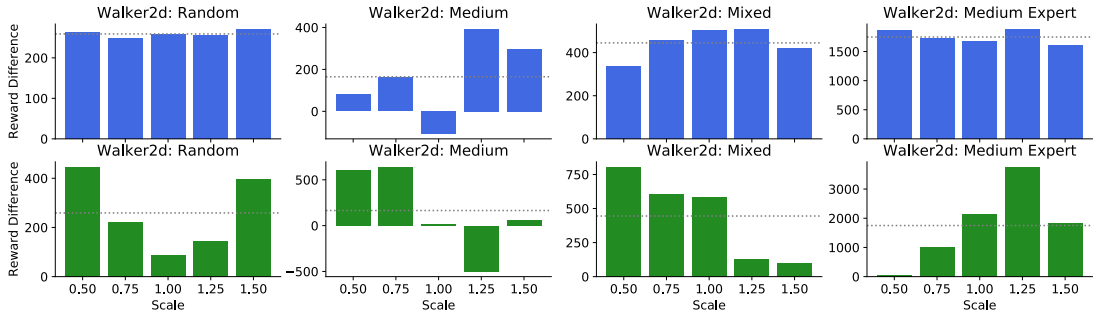
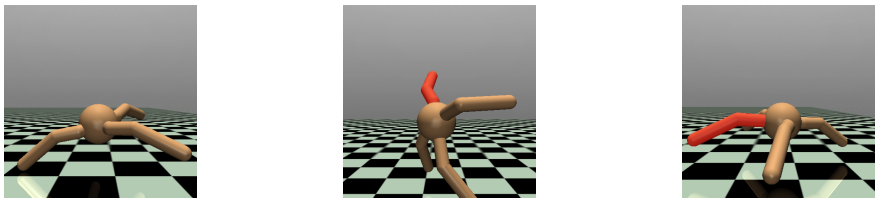


Figure 11. Mean improvement for Augmented World Models over MOPO for the Walker2d environment, averaged over five seeds. Top row (blue) = damping scale, bottom row (green) = mass scale. Dotted line is the mean, the same value for both damping and mass.



(a) MOPO on the regular Ant environment

(b) MOPO on the crippled leg Ant environment

(c) AugWM on the crippled leg Ant environment

Figure 12. Comparison between MOPO and AugWM on the crippled leg Ant environment. The crippled leg is highlighted in red. The MOPO agent flips over while the AugWM agent is able to adapt to changed dynamics and successfully run. MOPO on the regular Ant environment is provided as a reference.

Discussion We believe that our experiments provide significant support to the claim that training with AugWM improves zero-shot dynamics generalization. In a broad set of commonly used datasets, and with a wide range of out-of-distribution dynamics, our algorithm learns good policies where a state-of-the-art baseline fails.[3] This is due to a number of novel contributions: 1) using *dynamics augmentation* rather than *observation augmentation*; 2) training and testing with a context-based policy. Regarding limitations, we note that training inside the WM with context generally takes longer to converge (Appendix B). Furthermore, in more nonlinear settings such as the HalfCheetah modified body part setting, we saw a reduced performance for the learned context. This could be because the dynamics changes are out of the distribution of DAS augmentations (violating Eqn. 1), or due to the difficulty of modeling the task with a linear model. We note that linear models have achieved success in planning Gu et al. (2016) and meta learning Peng et al. (2021), and are effective in our case due to their data efficiency, but can be replaced by more flexible models to deal with different augmentations. Indeed, given our work is the first of its kind, we believe significant improvements are possible, such as using more complex and problem-specific augmentations.

[3]For videos see: https://sites.google.com/view/augmentedworldmodels/

6. Conclusion and Future Work

In this paper we propose Augmented World Models (AugWM), which we show sufficiently simulates changes in dynamics such that agents can generalize in a zero-shot manner. We believe that we are the first to propose this problem setting, and our results show a significant improvement over existing state-of-the-art methods which ignore this problem.

A promising line of future work would be to meta-train a policy over AugWM such that it can quickly adapt to new dynamics in the few-shot setting. There is evidence that data augmentation can improve robustness in meta-learning Rajendran et al. (2020), and could extend to strong performance in out-of-distribution tasks. We also wish to consider varying goals at test time, and other potential sources of non-stationarity which may impact policy performance Igl et al. (2021). It may also be possible to extend AugWM to pixel-based tasks, which have received a great deal of recent attention Hafner et al. (2019; 2020). We believe that our transition based augmentations will be applicable to a latent representation, as commonly used in state-of-the-art vision MBRL approaches. Thus, we think that extending our work to this setting, while a considerable feat of engineering, should not require significant methodological changes.

Acknowledgments

Philip Ball is funded through the Willowgrove Studentship. Cong Lu is funded by the Engineering and Physical Sciences Research Council (EPSRC). We are grateful to Taylor Killian for useful discussions on contextual/HiP-MDPs, and to Vitaly Kurin for his feedback on an earlier version of this paper via his 'Paper Notes'. The authors would also like to thank the anonymous ICLR SSL-RL Workshop + ICML reviewers and the area chair for constructive feedback which helped us in improving the paper.

Changes From ICML 2021 Proceedings

We added additional related work that we were not originally aware of, updated the Acknowledgments section, and generally tidied up the formatting.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. (2021). OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*.

Andrychowicz, O. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Antonova, R., Cruciani, S., Smith, C., and Kragic, D. (2017). Reinforcement learning for pivoting task.

Argenson, A. and Dulac-Arnold, G. (2021). Model-based offline planning. In *International Conference on Learning Representations*.

Ball, P., Parker-Holder, J., Pacchiano, A., Choromanski, K., and Roberts, S. (2020). Ready policy one: World building through active learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31*, pages 4754–4765.

Clavera, I., Nagabandi, A., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. (2019). Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. (2018). Model-based reinforcement learning via meta-policy optimization. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 617–629. PMLR.

Doshi-Velez, F. and Konidaris, G. (2016). Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1432–1440. AAAI Press.

Dulac-Arnold, G., Mankowitz, D. J., and Hester, T. (2019). Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2021). D4{rl}: Datasets for deep data-driven reinforcement learning.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2829–2838. PMLR.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NeurIPS'18, pages 2455–2467.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2020). Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, pages 2555–2565.

Hallak, A., Castro, D. D., and Mannor, S. (2015). Contextual markov decision processes.

Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A. A., Pinto, L., and Wang, X. (2021). Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*.

Henderson, P., Chang, W.-D., Shkurti, F., Hansen, J., Meger, D., and Dudek, G. (2017). Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. (2021). Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*.

James, S., Davison, A. J., and Johns, E. (2017). Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *1st Conference on Robot Learning*.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*.

Kaiser, L., Babaeizadeh, M., Milos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. (2020). Model based reinforcement learning for Atari. In *International Conference on Learning Representations*.

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

Killian, T., Konidaris, G., and Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden parameter markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Kostrikov, I., Yarats, D., and Fergus, R. (2021). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*.

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

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020a). Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems 33*.

Laskin, M., Srinivas, A., and Abbeel, P. (2020b). CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems.

Modi, A., Jiang, N., Singh, S., and Tewari, A. (2018). Markov decision processes with continuous side information. In *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pages 597–618. PMLR.

Nagabandi, A., Finn, C., and Levine, S. (2019). Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60 vol.1.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. (2019). Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113.

Peng, M., Zhu, B., and Jiao, J. (2021). Linear representation meta-reinforcement learning for instant adaptation. *CoRR*.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation, ICRA*.

Pitis, S., Creager, E., and Garg, A. (2020). Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*.

Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., and Fergus, R. (2020). Automatic data augmentation for generalization in deep reinforcement learning. *CoRR*, abs/2006.12862.

Rajendran, J., Irpan, A., and Jang, E. (2020). Meta-learning requires meta-augmentation. In *Advances in Neural Information Processing Systems*.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pages 5331–5340. PMLR.

Rudner, T. G. J., Lu, C., Osborne, M. A., Gal, Y., and Teh, Y. W. (2021). On pathologies in kl-regularized reinforcement learning from expert demonstrations. *ICLR RobustML Workshop*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. (2019). Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *CoRR*, abs/1911.08265.

Seo, Y., Lee, K., Clavera, I., Kurutach, T., Shin, J., and Abbeel, P. (2020). Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. (2020). Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, volume 31, pages 5334–5344. Curran Associates, Inc.

Wellmer, Z. and Kwok, J. T. (2021). Dropout's dream land: Generalization from learned simulators to reality. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

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

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*.

Yu, W., Tan, J., Liu, C. K., and Turk, G. (2017). Preparing for the unknown: Learning a universal policy with online system identification. In Amato, N. M., Srinivasa, S. S., Ayanian, N., and Kuindersma, S., editors, *Robotics: Science and Systems XIII*.

Zhang, A., Sodhani, S., Khetarpal, K., and Pineau, J. (2021). Learning robust state abstractions for hidden-parameter block mdps. In *International Conference on Learning Representations*.

Zhou, W., Pinto, L., and Gupta, A. (2019). Environment probing interaction policies. In *International Conference on Learning Representations*.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. (2020). Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*.

Zintgraf, L. M., Feng, L., Lu, C., Igl, M., Hartikainen, K., Hofmann, K., and Whiteson, S. (2021). Exploration in approximate hyper-state space for meta reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12991–13001. PMLR.

Limitations and Future Work

One of the primary limitations of Augmented World Models is what tasks our augmentations are suitable for, with the paper showcasing applications in modified physics and robot geometry. However, scaling dynamics could be less suitable for generalizing to external forces such as wind or more extreme changes like operating in no gravity at all. One potential richer class of structured augmentations that could account for these could be affine transformations that additionally add translations to the state. Furthermore, while the environments we consider in the paper use proprioceptive observations, we hypothesize that our work could be extended to environments with high-dimensional observations by performing augmentations in the latent space of a learned dynamics model (Hafner et al., 2020a,b). An even more exciting prospect could be to leverage recent video-language world models such as GAIA-1 (Hu et al., 2023) which can condition on combinations of initial frames, actions, or even text descriptions of the road conditions like "it is snowing" or "it is foggy". In this way, we could switch from dynamics augmentations to *semantic augmentations* that target more broad modes of real-world test-time changes. As such, we believe our algorithm represents early steps towards generalist agents that can hypothesize realistic unseen situations at deployment and be robust to them.

Next, we describe various algorithmic improvements we could make to AugWM. In the paper, we considered the relative underperformance of the baseline under changed dynamics (Figure 5), and ways to mitigate this. While some dynamics settings may admit higher potential return, e.g. lower mass, others may have a lower return ceiling. A metric that takes this into account for each task is *regret*, or the difference between the optimal and the attained return. Related to this idea is regret-based sampling (Dennis et al., 2021; Parker-Holder et al., 2022), which prefers to choose tasks that are challenging to the current agent. This could greatly improve on the uniform sampling scheme we have in our paper.

Another interesting target for improving dynamics generalization is the test-time context inference. At the moment, our approach trains a simple linear model in order to recover the approximate context. A more direct way to recover the correct context could be to rescale the dynamics model in the same way as the DAS augmentation, fitted to the test-time observations. Both methods would however be passively inferring the context and would not directly model information-gathering behavior. To allow for this, we could use similar techniques as in meta-RL (Hausknecht and Stone, 2015; Zintgraf et al., 2021a,b) which learn to optimally trade-off task inference versus reward. Since these methods usually need to observe a sequence of observations to infer the task (e.g. using an RNN), we would also need to ensure the dynamics model would remain accurate for long enough under augmentations.

Finally, while our paper only considered dynamics generalization, meta-RL (Rakelly et al., 2019b; Zintgraf et al., 2021a,b) often considers changing goals or reward at test-time. In the same way that we defined a distribution over plausible MDPs with varying dynamics by applying structured augmentations, we could also define a training distribution over MDPs with varying reward. We describe here one possible way to do so. Since the learned reward function is a neural network, it is linear over the representation learned in the penultimate layer. Concretely, $R(s,a) = w^\top \phi(s,a)$ where $\phi(s,a)$ is the output from the penultimate layer of the reward network. Varying the weight vector w could naturally lead to a family of linear reward functions from a single environment. This is a similar parameterization to successor features (Barreto et al., 2017; Borsa et al., 2018; Filos et al., 2021) but successor features have typically been applied to settings where examples of multiple reward functions and goals exist.

4

Synthetic Experience Replay

In the previous chapter, we investigated synthetic data generation using learned forward dynamics models (Hafner et al., 2020b; Janner et al., 2019; Yu et al., 2020b), i.e. those that learn a conditional distribution $P_\psi(s', r|s, a)$ and bootstrap rollouts from real states s. These models are nearly ubiquitous in modern model-based reinforcement learning but suffer from two conceptual pain points—first, data generation bootstraps from real states leading to limited generalization at the beginning; and second, learned models suffer from compounding error over time which precludes the use of long model rollouts (Janner et al., 2019). Indeed, the original Dyna (Sutton, 1991) framework introduced in Section 2.2.1 made few of these assumptions, but instead proposed a more general framework for integrating synthetic environment transitions. In the same spirit, we propose a conceptually simple and novel approach to synthetic data generation, *Synthetic Experience Replay* (SynthER), which uses generative modeling to directly model and upsample (or increase the quantity of) the training data of an agent.

In the past few years, diffusion models (Ho et al., 2020; Karras et al., 2022) have taken the world by storm, resolving many of the issues that plagued earlier generative models like mode-collapse in VAEs (Kingma and Welling, 2014) and unstable

training in GANs (Goodfellow et al., 2014). Diffusion models are particularly strong at ensuring coverage over the data distribution while maintaining individual sample quality; this makes them ideal for modeling reinforcement learning data. To confirm this hypothesis, we show that synthetic data generated from the offline D4RL (Fu et al., 2020) benchmark closely reproduces the statistics of the original data.

In the empirical evaluation, we continue on the D4RL benchmark and show that the synthetic data overall faithfully reproduces the original performance for a wide selection of offline RL algorithms. More notably, we also see particularly strong results for upsampling reduced variants of the D4RL datasets, matching the original performance *starting from as little as 3% of the original data*. This strongly improves on baseline data augmentation approaches, and we show that the data we generate is simultaneously more diverse and more faithful to the true dynamics of the environment. Furthermore, we show that additional synthetic data from SynthER allows us to greatly increase the network size in the TD3+BC (Fujimoto and Gu, 2021) algorithm leading to a +17.7% overall gain on D4RL.

Next, we show that when upsampling data online, we can improve the data efficiency of the standard Soft Actor-Critic (Haarnoja et al., 2018) algorithm beyond specially designed data-efficient algorithms (Chen et al., 2021b) on a range of OpenAI Gym (Brockman et al., 2016) and Deepmind Control Suite (Tassa et al., 2020) environments, *without any algorithmic changes*. Finally, we show that our method extends to high-dimensional pixel-based environments by generating data in the latent space of a pre-trained encoder. By evaluating this latent data with the offline DrQ+BC (Lu et al., 2022b) algorithm, we demonstrate an average +9.5% improvement on the offline V-D4RL (Lu et al., 2022b) benchmark. Since DrQ+BC uses random crop data augmentations on the raw images, this result shows that our method can be used in conjunction with standard data augmentation techniques.

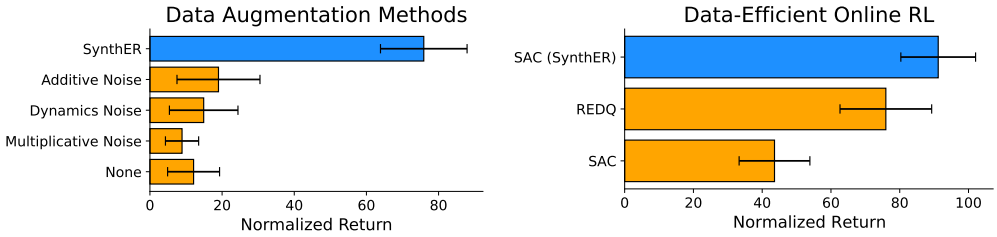
Cong Lu*, Philip J. Ball*, Yee Whye Teh, and Jack Parker-Holder. Synthetic Experience Replay. **In NeurIPS, 2023.**

Synthetic Experience Replay

Cong Lu*, **Philip J. Ball***, **Yee Whye Teh, Jack Parker-Holder**
University of Oxford

Abstract

A key theme in the past decade has been that when large neural networks and large datasets combine they can produce remarkable results. In deep reinforcement learning (RL), this paradigm is commonly made possible through *experience replay*, whereby a dataset of past experiences is used to train a policy or value function. However, unlike in supervised or self-supervised learning, an RL agent has to collect its own data, which is often limited. Thus, it is challenging to reap the benefits of deep learning, and even small neural networks can overfit at the start of training. In this work, we leverage the tremendous recent progress in generative modeling and propose Synthetic Experience Replay (SYNTHER), a diffusion-based approach to flexibly upsample an agent's collected experience. We show that SYNTHER is an effective method for training RL agents across offline and online settings, in both proprioceptive and pixel-based environments. In offline settings, we observe drastic improvements when upsampling small offline datasets and see that additional synthetic data also allows us to effectively train larger networks. Furthermore, SYNTHER enables online agents to train with a much higher update-to-data ratio than before, leading to a significant increase in sample efficiency, *without any algorithmic changes*. We believe that synthetic training data could open the door to realizing the full potential of deep learning for replay-based RL algorithms from limited data. Finally, we open-source our code at `https://github.com/conglu1997/SynthER`.



(a) IQL [41] on a reduced 15% subset of walker2d medium-replay [21].

(b) SAC [24] on 6 DeepMind Control Suite [71] and OpenAI Gym [10] environments.

Figure 1: Upsampling data using SYNTHER greatly outperforms explicit data augmentation schemes for small offline datasets and data-efficient algorithms in online RL *without any algorithmic changes*. Moreover, synthetic data from SYNTHER may readily be added to *any* algorithm utilizing experience replay. Full results in Section 4.

1 Introduction

In the past decade, the combination of large datasets [14, 63] and ever deeper neural networks [15, 25, 43, 73] has led to a series of more generally capable models [11, 56, 58]. In reinforcement learning (RL, Sutton and Barto [67]), agents typically learn online from their own experience. Thus,

*Equal contribution. Correspondence to `cong.lu@stats.ox.ac.uk` and `ball@robots.ox.ac.uk`.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

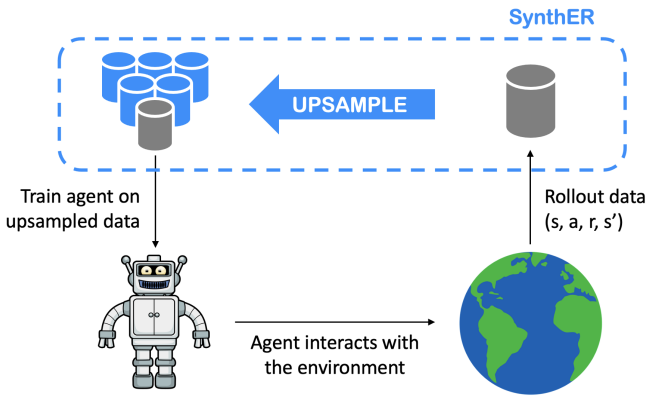


Figure 2: SYNTHER allows any RL agent using experience replay to arbitrarily upsample, or increase the quantity of, their experiences (in **grey**) and train on synthetic data (in **blue**). We evaluate our approach up to a factor of $100\times$ more data in Section 4.2, across both proprioceptive and pixel-based environments. By leveraging this increased data, agents can learn effectively from smaller datasets and achieve higher sample efficiency. Details of the upsampling process are given in Figure 3.

to leverage sufficiently rich datasets, RL agents typically make use of *experience replay* [19, 53], where training takes place on a dataset of recent experiences. However, this experience is typically limited, unless an agent is distributed over many workers which requires both high computational cost and sufficiently fast simulation [18, 37].

Another approach for leveraging broad datasets for training RL policies is *offline* RL [2, 46], whereby behaviors may be distilled from previously collected data either via behavior cloning [61], off-policy learning [22, 44] or model-based methods [39, 49, 78]. Offline data can also significantly bootstrap online learning [9, 27, 75]; however, it is a challenge to apply these methods when there is a mismatch between offline data and online environment. Thus, many of the successes rely on toy domains with transfer from specific behaviors in a simple low-dimensional proprioceptive environment.

Whilst strong results have been observed in re-using prior data in RL, appropriate data for particular behaviors may simply not exist and thus this approach falls short in generality. We consider an alternative approach—rather than passively reusing data, we leverage tremendous progress in generative modeling to generate a large quantity of new, synthetic data. While prior work has considered upsampling online RL data with VAEs or GANs [32, 34, 51], we propose making use of *diffusion* generative models [30, 38, 66], which unlocks significant new capabilities.

Our approach, which we call *Synthetic Experience Replay*, or SYNTHER, is conceptually simple, whereby given a limited initial dataset, we can arbitrarily upsample the data for an agent to use as if it was real experience. Therefore, in this paper, we seek to answer a simple question: *Can the latest generative models replace or augment traditional datasets in reinforcement learning?* To answer this, we consider the following settings: offline RL where we entirely replace the original data with data produced by a generative model, and online RL where we upsample experiences to broaden the training data available to the agent. In both cases, SYNTHER leads to drastic improvements, obtaining performance comparable to that of agents trained with substantially more real data. Furthermore, in certain offline settings, synthetic data enables effective training of larger policy and value networks, resulting in higher performance by alleviating the representational bottleneck. Finally, we show that SYNTHER scales to pixel-based environments *by generating data in latent space*. We thus believe this paper presents sufficient evidence that our approach could enable entirely new, efficient, and scalable training strategies for RL agents. To summarize, the contributions of this paper are:

- We propose SYNTHER in Section 3, a diffusion-based approach that allows one to generate synthetic experiences and thus arbitrarily upsample data for any reinforcement learning algorithm utilizing experience replay.
- We validate the synthetic data generated by SYNTHER in offline settings across proprioceptive and pixel-based environments in Section 4.1 and Section 4.3, presenting the first generative approach to show parity with real data on the standard D4RL and V-D4RL offline datasets with a wide variety of algorithms. Furthermore, we observe considerable improvements from upsampling for small offline datasets and scaling up network sizes.
- We show how SYNTHER can arbitrarily upsample an online agent's training data in Section 4.2 by continually training the diffusion model. This allows us to significantly increase an agent's update-to-data (UTD) ratio matching the efficiency of specially designed data-efficient algorithms *without any algorithmic changes*.

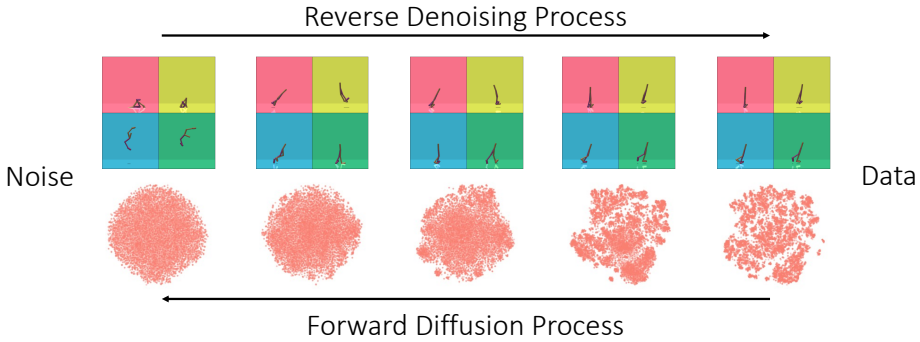


Figure 3: SYNTHER generates synthetic samples using a diffusion model which we visualize on the proprioceptive walker2d environment. On the **top row**, we render the state component of the transition tuple on a subset of samples; and on the **bottom row**, we visualize a t-SNE [72] projection of 100,000 samples. The denoising process creates cohesive and plausible transitions whilst also remaining diverse, as seen by the multiple clusters that form at the end of the process in the bottom row.

2 Background

2.1 Reinforcement Learning

We model the environment as a Markov Decision Process (MDP, Sutton and Barto [67]), defined as a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(s'|s, a)$ the transition dynamics, $R(s, a)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in (0, 1)$ the discount factor. The goal in reinforcement learning is to optimize a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

2.2 Offline Reinforcement Learning

In *offline RL* [46], the policy is not deployed in the environment until test time. Instead, the algorithm only has access to a static dataset $\mathcal{D}_{\text{env}} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$, collected by one or more behavioral policies π_b. We refer to the distribution from which $\mathcal{D}_{\text{env}}$ was sampled as the *behavioral distribution* [78]. In some of the environments we consider, the environment may be finite horizon or have early termination. In that case, the transition tuple also contains a terminal flag d_t where $d_t = 1$ indicates the episode ended early at timestep t and $d_t = 0$ otherwise.

2.3 Diffusion Models

Diffusion models [30, 66] are a class of generative models inspired by non-equilibrium thermodynamics that learn to iteratively reverse a forward noising process and generate samples from noise. Given a data distribution $p(\mathbf{x})$ with standard deviation σ_{data}, we consider noised distributions $p(\mathbf{x}; \sigma)$ obtained by adding i.i.d. Gaussian noise of standard deviation σ to the base distribution. The forward noising process is defined by a sequence of noised distributions following a fixed noise schedule $\sigma_0 = \sigma_{\max} > \sigma_1 > \cdots > \sigma_N = 0$. When $\sigma_{\max} \gg \sigma_{\text{data}}$, the final noised distribution $p(\mathbf{x}; \sigma_{\max})$ is essentially indistinguishable from random noise.

Karras et al. [38] consider a probability-flow ODE with the corresponding continuous noise schedule $\sigma(t)$ that maintains the desired distribution as $\mathbf{x}$ evolves through time given by Equation (1).

$$\mathrm{d}\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))\mathrm{d}t \tag{1}$$

where the dot indicates a time derivative and $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))$ is the score function [33], which points towards the data at a given noise level. Infinitesimal forward or backward steps of this ODE either nudge a sample away or towards the data. Karras et al. [38] consider training a denoiser $D_\theta(\mathbf{x}; \sigma)$ on an L2 denoising objective:

$$\min_\theta \mathbb{E}_{\mathbf{x}\sim p, \sigma, \epsilon\sim\mathcal{N}(0,\sigma^2 I)} \|D_\theta(\mathbf{x} + \epsilon; \sigma) - \mathbf{x}\|_2^2 \tag{2}$$

and then use the connection between score-matching and denoising [74] to obtain $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) = (D_\theta(\mathbf{x}; \sigma) - \mathbf{x})/\sigma^2$. We may then apply an ODE (or SDE as a generalization of Equation (1)) solver to reverse the forward process. In this paper, we train our diffusion models to approximate the online or offline behavioral distribution.

Algorithm 1 SYNTHER for online replay-based algorithms. Our additions are highlighted in **blue**.

Input: real data ratio $r \in [0, 1]$
Initialize: $\mathcal{D}_{\text{real}} = \emptyset$ real replay buffer, π agent, $\mathcal{D}_{\text{synthetic}} = \emptyset$ synthetic replay buffer, M diffusion model
for $t = 1, \ldots, T$ **do**
    Collect data with π in the environment and add them to $\mathcal{D}_{\text{real}}$
    Update diffusion model M with samples from $\mathcal{D}_{\text{real}}$
    Generate samples from M and add them to $\mathcal{D}_{\text{synthetic}}$
    Train π on samples from $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synthetic}}$ mixed with ratio r
end for

3 Synthetic Experience Replay

In this section, we introduce Synthetic Experience Replay (SYNTHER), our approach to upsampling an agent's experience using diffusion. We begin by describing the simpler process used for offline RL and then how that may be adapted to the online setting by continually training the diffusion model.

3.1 Offline SYNTHER

For offline reinforcement learning, we take the data distribution of the diffusion model $p(\mathbf{x})$ to simply be the offline behavioral distribution. In the proprioceptive environments we consider, the full transition is low-dimensional compared with typical pixel-based diffusion. Therefore, the network architecture is an important design choice; and similarly to Pearce et al. [55] we find it important to use a residual MLP denoising [70] network. Furthermore, the choice of the Karras et al. [38] sampler allows us to use a low number of diffusion steps ($n = 128$) resulting in high sampling speed. Full details for both are provided in Appendix B. We visualize the denoising process on a representative D4RL [21] offline dataset, in Figure 3. We further validate our diffusion model on the D4RL datasets in Figure 8 in Appendix A by showing that the synthetic data closely matches the original data when comparing the marginal distribution over each dimension. In Section 4.3, we show the same model may be used for pixel-based environments by generating data in a low-dimensional latent space.

Next, we conduct a quantitative analysis and show that **the quality of the samples from the diffusion model is significantly better** than with prior generative models such as VAEs [40] and GANs [23]. We consider the state-of-the-art Tabular VAE (TVAE) and Conditional Tabular GAN (CTGAN) models proposed by Xu et al. [76], and tune them on the D4RL halfcheetah medium-replay dataset. Full hyperparameters are given in Appendix B.1. As proposed in Patki et al. [54], we compare the following two high-level statistics: **(1) Marginal:** Mean Kolmogorov-Smirnov [52] statistic, measuring the maximum distance between empirical cumulative distribution functions, for each dimension of the synthetic and real data; and **(2) Correlation:** Mean Correlation Similarity, measuring the difference in pairwise Pearson rank correlations [20] between the synthetic and real data.

We also assess downstream offline RL performance using the synthetic data with two state-of-the-art offline RL algorithms, TD3+BC [22] and IQL [41], in Table 1. The full evaluation protocol is described in Section 4.1. The diffusion model is far more faithful to the original data than prior generative models which leads to substantially higher returns on both algorithms. Thus, we hypothesize a large part of the failure of prior methods [34, 51] is due to a weaker generative model.

3.2 Online SYNTHER

SYNTHER may be used to upsample an online agent's experiences by continually training the diffusion model on new experiences. We provide pseudocode for how to incorporate SYNTHER

Table 1: SYNTHER is better at capturing both the high-level statistics of the dataset (halfcheetah medium-replay) than prior generative models and also leads to far higher downstream performance. Metrics (left) computed from 100K samples from each model, offline RL performance (right) computed using 5M samples from each model. We show the mean and standard deviation of the final performance averaged over 8 seeds.

Model	Metrics		Eval. Return	
	Marginal	Correlation	TD3+BC	IQL
Diffusion (Ours)	**0.989**	**0.998**	**45.9±0.9**	**46.6±0.2**
VAE [76]	0.942	0.979	27.1±2.1	15.2±2.2
GAN [76]	0.959	0.981	24.3±1.9	15.9±2.4

Table 2: A comprehensive evaluation of SYNTHER on a wide variety of proprioceptive D4RL [21] datasets and selection of state-of-the-art offline RL algorithms. We show that synthetic data from SYNTHER faithfully reproduces the original performance, which allows us to completely eschew the original training data. We show the mean and standard deviation of the final performance averaged over 8 seeds. **Highlighted** figures show at least parity over each group (algorithm and environment class) of results.

Environment	Behavioral Policy	TD3+BC [22]		IQL [41]		EDAC [5]	
		Original	SYNTHER	Original	SYNTHER	Original	SYNTHER
halfcheetah-v2	random	11.3±0.8	12.2±1.1	15.2±1.2	17.2±3.4		-
	mixed	44.8±0.7	45.9±0.9	43.5±0.4	46.6±0.2	62.1±1.3	63.0±1.3
	medium	48.1±0.2	49.9±1.2	48.3±0.1	49.6±0.3	67.7±1.2	65.1±1.3
	medexp	90.8±7.0	87.2±11.1	94.6±0.2	93.3±2.6	104.8±0.7	94.1±10.1
walker2d-v2	random	0.6±0.3	2.3±1.9	4.1±0.8	4.2±0.3		-
	mixed	85.6±4.6	90.5±4.3	82.6±8.0	83.3±5.9	87.1±3.2	89.8±1.5
	medium	82.7±5.5	84.8±1.4	84.0±5.4	84.7±5.5	93.4±1.6	93.4±2.4
	medexp	110.0±0.4	110.2±0.5	111.7±0.6	111.4±0.7	114.8±0.9	114.7±1.2
hopper-v2	random	8.6±0.3	14.6±9.4	7.2±0.2	7.7±0.1		-
	mixed	64.4±24.8	53.4±15.5	84.6±13.5	103.2±0.4	99.7±0.9	101.4±0.8
	medium	60.4±4.0	63.4±4.2	62.8±6.0	72.0±4.5	101.7±0.3	102.4±0.5
	medexp	101.1±10.5	105.4±9.7	106.2±6.1	90.8±17.9	105.2±11.6	109.7±0.2
locomotion average		**59.0±4.9**	**60.0±5.1**	**62.1±3.5**	**63.7±3.5**	**92.9±2.4**	**92.6±2.1**
maze2d-v1	umaze	29.4±14.2	37.6±14.4	37.7±2.0	41.0±0.7	95.3±7.4	99.1±18.6
	medium	59.5±41.9	65.2±36.1	35.5±1.0	35.1±2.6	57.0±4.0	66.4±10.9
	large	97.1±29.3	92.5±38.5	49.6±22.0	60.8±5.3	95.6±26.5	143.3±21.7
maze average		**62.0±28.2**	**65.1±29.7**	**40.9±8.3**	**45.6±2.9**	**82.6±12.6**	**102.9±17.1**

into any online replay-based RL agent in Algorithm 1 and visualize this in Figure 2. Concretely, a diffusion model is periodically updated on the real transitions and then used to populate a second synthetic buffer. The agent may then be trained on a mixture of real and synthetic data sampled with ratio r. For the results in Section 4.2, we simply set $r = 0.5$ following Ball et al. [9]. The synthetic replay buffer may also be configured with a finite capacity to prevent overly stale data.

4 Empirical Evaluation

We evaluate SYNTHER across a wide variety of offline and online settings. First, we validate our approach on offline RL, where we entirely replace the original data, and further show large benefits from upsampling small offline datasets. Next, we show that SYNTHER leads to large improvements in sample efficiency in online RL, exceeding specially designed data-efficient approaches. Furthermore, we show that SYNTHER scales to pixel-based environments by generating data in latent space. Finally, we perform a meta-analysis over our empirical evaluation using the RLiable [3] framework in Figure 7.

4.1 Offline Evaluation

We first verify that synthetic samples from SYNTHER faithfully model the underlying distribution from the canonical offline D4RL [21] datasets. To do this, we evaluate SYNTHER in combination with 3 widely-used SOTA offline RL algorithms: TD3+BC (Fujimoto and Gu [22], explicit policy regularization), IQL (Kostrikov et al. [41], expectile regression), and EDAC (An et al. [5], uncertainty-based regularization) on an extensive selection of D4RL datasets. We consider the MuJoCo [69] locomotion (halfcheetah, walker2d, and hopper) and maze2d environments. In these experiments, all datasets share the same training hyperparameters in Appendix B, with some larger datasets using a wider network. For each dataset, we upsample the original dataset to **5M samples**; we justify this choice in Appendix C.1. We show the final performance in Table 2.

Our results show that we achieve at least parity for all groups of environments and algorithms as highlighted in the table, *regardless of the precise details of each algorithm*. We note significant improvements to maze2d environments, which are close to the 'best' performance as reported in CORL [68] (i.e., the best iteration during offline training) rather than the final performance. We hypothesize this improvement is largely due to increased data from SYNTHER, which leads to less overfitting and increased stability. For the locomotion datasets, we largely reproduce the original results, which we attribute to the fact that most D4RL datasets are at least 1M in size and are already sufficiently large. However, as detailed in Table 5 in Appendix A.1, SYNTHER allows the effective size of the dataset to be compressed significantly, up to $12.9\times$ on some datasets. Finally, we present results on the AntMaze environment in Appendix E.1, and experiments showing that the synthetic and real data are compatible with each other in Appendix E.2.

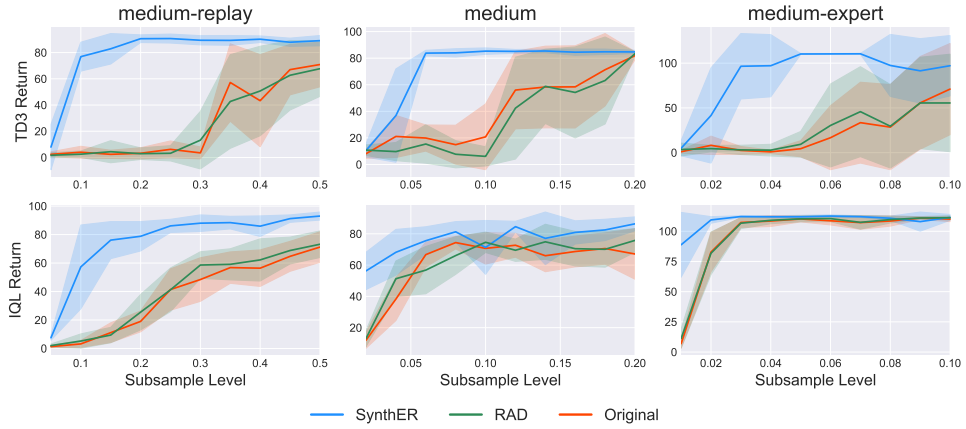


Figure 4: SYNTHER is a powerful method for upsampling reduced variants of the walker2d datasets and vastly improves on competitive explicit data augmentation approaches for both the TD3+BC (top) and IQL (bottom) algorithms. The subsampling levels are scaled proportionally to the original size of each dataset. We show the mean and standard deviation of the final performance averaged over 8 seeds.

4.1.1 Upsampling for Small Datasets

We investigate the benefit of SYNTHER for small offline datasets and compare it to canonical 'explicit' data augmentation approaches [8, 45]. Concretely, we wish to understand whether SYNTHER generalizes and generates synthetic samples that improve policy learning compared with *explicitly* augmenting the data with hand-designed inductive biases. We focus on the walker2d (medium, medium-replay/mixed, medium-expert) datasets in D4RL and uniformly subsample each at the transition level. We subsample each dataset proportional to the original dataset size so that the subsampled datasets approximately range from 20K to 200K samples. As in Section 4.1, we then use SYNTHER to *upsample* each dataset to 5M transitions. Our denoising network uses the same hyperparameters as for the original evaluation in Section 4.1.

In Figure 4, we can see that for all datasets, SYNTHER leads to a significant gain in performance and vastly improves on explicit data augmentation approaches. For explicit data augmentation, we select the overall most effective augmentation scheme from Laskin et al. [45] (adding Gaussian noise of the form $\epsilon \sim \mathcal{N}(0, 0.1)$). Notably, with SYNTHER we can achieve close to the original levels of performance on the walker2d-medium-expert datasets starting from **only 3% of the original data**. In Figure 1a, we methodically compare across both additive and multiplicative versions of RAD, as well as dynamics augmentation [8] on the 15% reduced walker medium-replay dataset.

Why is SYNTHER better than explicit augmentation? To provide intuition into the efficacy of SYNTHER over canonical explicit augmentation approaches, we compare the data generated by SYNTHER to that generated by the best-performing data augmentation approach in Figure 1a, namely additive noise. We wish to evaluate two properties: 1) How diverse is the data? 2) How accurate is the data for the purposes of learning policies? To measure diversity, we measure the *minimum* L2 distance of each datapoint from the dataset, which allows us to see how far the upsampled data is from the original data. To measure the validity of the data, we follow Lu et al. [49] and measure the MSE between the reward and next state proposed by SYNTHER with the true next state and reward defined by the simulator. We plot both these values in a joint scatter plot to compare how they vary with respect to each other. For this, we compare specifically on the reduced 15% subset of walker2d medium-replay as in Figure 1a. As we see in Figure 5, SYNTHER generates a significantly wider marginal distribution over the distance from the dataset, and generally produces samples that are further away from the dataset than explicit augmentations. Remarkably, however, we see that these samples are far more consistent with the true environment dynamics. Thus, SYNTHER

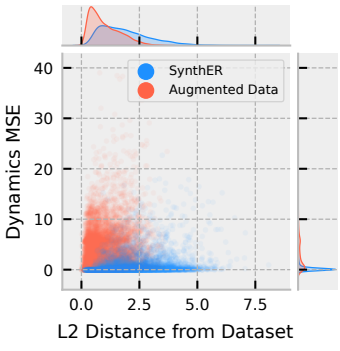


Figure 5: Comparing L2 distance from training data and dynamics accuracy under SYNTHER and augmentations.

Table 3: SYNTHER enables effective training of larger policy and value networks for TD3+BC [22] leading to a **11.7%** gain on the offline MuJoCo locomotion datasets. In comparison, simply increasing the network size with the original data does not improve performance. We show the mean and standard deviation of the final performance averaged over 8 seeds.

Environment	Behavioral Policy	Baseline	Larger Network	
			Original Data	SYNTHER
halfcheetah-v2	random	11.3±0.8	11.0±0.7	**12.8±0.8**
	mixed	44.8±0.7	44.8±1.1	**48.0±0.5**
	medium	48.1±0.2	50.2±1.8	**53.3±0.4**
	medexp	90.8±7.0	95.5±5.4	**100.1±2.7**
walker2d-v2	random	0.6±0.3	2.6±2.1	**4.3±1.7**
	mixed	85.6±4.6	76.4±9.9	**93.6±3.6**
	medium	82.7±5.5	84.5±1.7	**87.2±1.2**
	medexp	110.0±0.4	110.3±0.5	110.2±0.3
hopper-v2	random	8.6±0.3	10.3±5.6	19.5±11.2
	mixed	64.4±24.8	62.4±21.6	**86.8±12.8**
	medium	60.4±4.0	61.9±5.9	65.1±4.7
	medexp	101.1±10.5	104.6±9.4	**109.7±4.1**
locomotion average		59.0±4.9	59.5±5.5	**65.9±3.7**

generates samples that have significantly lower dynamics MSE than explicit augmentations, even for datapoints that are far away from the training data. This implies that a high level of generalization has been achieved by the SYNTHER model, resulting in the ability to generate **novel, diverse, yet dynamically accurate data** that can be used by policies to improve performance.

4.1.2 Scaling Network Size

A further benefit we observe from SYNTHER on the TD3+BC algorithm is that upsampled data can enable scaling of the policy and value networks leading to improved performance. As is typical for RL algorithms, TD3+BC uses a small value and policy network with two hidden layers, and width of 256, and a batch size of 256. We consider increasing the size of both networks to be three hidden layers and width 512 (approximately $6\times$ more parameters), and the batch size to 1024 to better make use of the upsampled data in Table 3.

We observe a large overall improvement of **11.7%** for the locomotion datasets when using a larger network with synthetic data (Larger Network + SYNTHER). Notably, when using the original data (Larger Network + Original Data), the larger network performs the same as the baseline. This suggests that the bottleneck in the algorithm lies in the representation capability of the neural network and *synthetic samples from* SYNTHER *enables effective training of the larger network*. This could alleviate the data requirements for scaling laws in reinforcement learning [1, 28]. However, for the IQL and EDAC algorithms, we did not observe an improvement by increasing the network size which suggests that the bottleneck there lies in the data or algorithm rather than the architecture.

4.2 Online Evaluation

Next, we show that SYNTHER can effectively upsample an online agent's continually collected experiences. In this section, we follow the sample-efficient RL literature [12, 16] and consider 3 environments from the DeepMind Control Suite (DMC, Tunyasuvunakool et al. [71]) (cheetah-run, quadruped-walk, and reacher-hard) and 3 environments the OpenAI Gym Suite [10] (walker2d, halfcheetah, and hopper). As in Chen et al. [12], D'Oro et al. [16], we choose the base algorithm to be Soft Actor-Critic (SAC, Haarnoja et al. [24]), a popular off-policy entropy-regularized algorithm, and benchmark against a SOTA sample-efficient variant of itself, 'Randomized Ensembled Double Q-Learning' (REDQ, Chen et al. [12]). REDQ uses an ensemble of 10 Q-functions and computes target values across a randomized subset of them during training. By default, SAC uses an update-to-data ratio of 1 (1 update for each transition collected); the modifications to SAC in REDQ enable this to be raised to 20. Our method, 'SAC (SYNTHER)', augments the training data by generating 1M new samples for every 10K real samples collected and samples them with a ratio $r = 0.5$. We then match REDQ and train with a UTD ratio of 20. We evaluate our algorithms over 200K online steps for the DMC environments and 100K for OpenAI Gym.

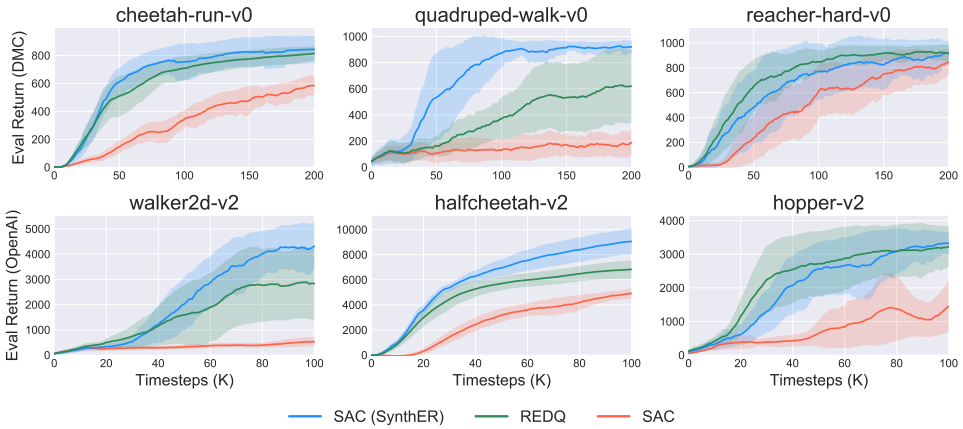


Figure 6: SYNTHER greatly improves the sample efficiency of online RL algorithms by enabling an agent to train on upsampled data. This allows an agent to use an increased update-to-data ratio (UTD=20 compared to 1 for regular SAC) *without any algorithmic changes*. We show the mean and standard deviation of the online return over 6 seeds. DeepMind Control Suite environments are shown in the top row, and OpenAI Gym environments are shown in the bottom.

In Figure 6, we see that SAC (SYNTHER) matches or outperforms REDQ on the majority of the environments with particularly strong results on the quadruped-walk and halfcheetah-v2 environments. This is particularly notable as D'Oro et al. [16] found that UTD=20 on average *decreased performance* for SAC compared with the default value of 1, attributable to issues with overestimation and overfitting [12, 48]. We aggregate the final performance on the environments in Figure 1b, normalizing the DMC returns following Lu et al. [50] and OpenAI returns as in D4RL. Moreover, due to the fast speed of training our diffusion models and fewer Q-networks, our approach is in fact faster than REDQ based on wall-clock time, whilst also requiring fewer algorithmic design choices, such as large ensembles and random subsetting. Full details on run-time are given in Appendix D.2.

4.3 Scaling to Pixel-Based Observations

Finally, we show that we can readily scale SYNTHER to pixel-based environments by generating data in the latent space of a CNN encoder. We consider the V-D4RL [50] benchmarking suite, a set of standardized pixel-based offline datasets, and focus on the 'cheetah-run' and 'walker-walk' environments. We use the associated DrQ+BC [50] and BC algorithms. Whilst the original image observations are of size $84 \times 84 \times 3$, we note that the CNN encoder in both algorithms generates features that are 50 dimensional [77]. Therefore, given a frozen encoder pre-trained on the same dataset, we can retain the fast training and sampling speed of our proprioceptive models but now in pixel space. We present full details in Appendix F.

Analogously to the proprioceptive offline evaluation in Section 4.1, we upsample 5M latent transitions for each dataset and present downstream performance in Table 4. Since the V-D4RL datasets are smaller than the D4RL equivalents with a base size of 100K, we would expect synthetic data to be beneficial. Indeed, we observe a statistically significant increase in performance of **+9.5%** and **+6.8%** on DrQ+BC and BC respectively; with particularly strong highlighted results on the medium and expert datasets. We believe this serves as compelling evidence of the scalability of SYNTHER to high-dimensional observation spaces and leave generating data in the original image space, or extending this approach to the online setting for future work.

5 Related Work

Whilst generative training data has been explored in reinforcement learning; in general, synthetic data has not previously performed as well as real data on standard RL benchmarks.

Generative Training Data. Imre [34], Ludjen [51] considered using VAEs and GANs to generate synthetic data for online reinforcement learning. However, we note that both works failed to match the original performance on simple environments such as CartPole—this is likely due to the use of a

Table 4: We scale SYNTHER to high dimensional pixel-based environments by generating data in the latent space of a CNN encoder pre-trained on the same offline data. Our approach is composable with algorithms that train with data augmentation and leads to a **+9.5%** and **+6.8%** overall gain on DrQ+BC and BC respectively. We show the mean and standard deviation of the final performance averaged over 6 seeds.

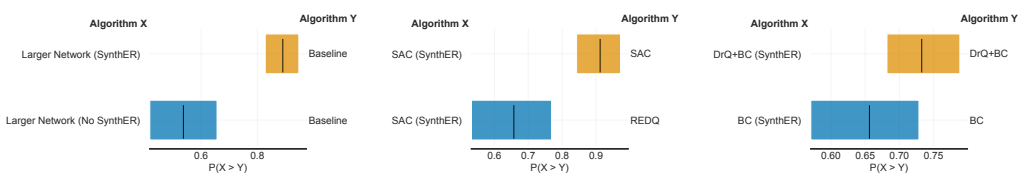
Environment		DrQ+BC [50]		BC [50]	
		Original	**SynthER**	**Original**	**SynthER**
walker-walk	mixed	28.7±6.9	32.3±7.6	16.5±4.3	12.3±3.6
	medium	46.8±2.3	44.0±2.9	40.9±3.1	40.3±3.0
	medexp	86.4±5.6	83.4±6.3	47.7±3.9	45.2±4.5
	expert	68.4±7.5	**83.6±7.5**	91.5±3.9	92.0±4.2
cheetah-run	mixed	44.8±3.6	43.8±2.7	25.0±3.6	27.9±3.4
	medium	53.0±3.0	56.0±1.2	51.6±1.4	52.2±1.2
	medexp	50.6±8.2	56.9±8.1	57.5±6.3	**69.9±9.5**
	expert	34.5±8.3	**52.3±7.0**	67.4±6.8	**85.4±3.1**
Average		51.7±5.7	56.5±5.4 **(+9.5%)**	49.8±4.2	53.2±4.1 **(+6.8%)**

weaker class of generative models which we explored in Section 3.1. Huang et al. [32] considered using GAN samples to *pre-train* an RL policy, observing a modest improvement in sample efficiency for CartPole. Chen et al. [13], Yu et al. [79] consider augmenting the image observations of robotic control data using a guided diffusion model whilst maintaining the same action. This differs from our approach which models the entire transition and *can synthesize novel action and reward labels*.

Outside of reinforcement learning, Azizi et al. [7], He et al. [26], Sariyildiz et al. [60] consider generative training data for image classification and pre-training. They also find that synthetic data improves performance for data-scarce settings which are especially prevalent in reinforcement learning. Sehwag et al. [64] consider generative training data to improve adversarial robustness in image classification. In continual learning, "generative replay" [65] has been considered to compress examples from past tasks to prevent forgetting.

Generative Modeling in RL. Prior work in diffusion modeling for offline RL has largely sought to supplant traditional reinforcement learning with "upside-down RL" [62]. Diffuser [36] models long sequences of transitions or full episodes and can bias the whole trajectory with guidance towards high reward or a particular goal. It then takes the first action and re-plans by receding horizon control. Decision Diffuser [4] similarly operates at the sequence level but instead uses conditional guidance on rewards and goals. Du et al. [17] present a similar trajectory-based algorithm for visual data. In contrast, SYNTHER operates at the transition level and seeks to be readily compatible with existing reinforcement learning algorithms. Pearce et al. [55] consider a diffusion-based approach to behavioral cloning, whereby a state-conditional diffusion model may be used to sample actions that imitate prior data. Azar et al. [6], Li et al. [47] provide theoretical sample complexity bounds for model-based reinforcement learning given access to a generative model.

Model-Based Reinforcement Learning. We note the parallels between our work and model-based reinforcement learning [35, 49, 78]; which tends to generate synthetic samples by rolling out using forward dynamics models. Two key differences of this approach to our method are: SYNTHER synthesizes new experiences without the need to start from a real state and the generated experiences are distributed exactly according to the data, rather than subject to compounding errors due to modeling inaccuracy. Furthermore, SYNTHER is an orthogonal approach which could in fact be *combined with* forward dynamics models by generating initial states using diffusion, which could lead to increased diversity.



(a) Offline TD3+BC Larger Networks (Full results in Table 3)

(b) Online DMC Evaluation (Full results in Figure 6)

(c) Offline V-D4RL Evaluation (Full results in Table 4)

Figure 7: RLiable [3] analysis allowing us to aggregate results across environments and show the probability of improvement for SYNTHER across our empirical evaluation.

6 Conclusion

In this paper, we proposed SYNTHER, a powerful and general method for upsampling agent experiences in any reinforcement learning algorithm using experience replay. We integrated SYNTHER with ease on **six distinct algorithms across proprioceptive and pixel-based environments**, each fine-tuned for its own use case, **with no algorithmic modification**. Our results show the potential of synthetic training data when combined with modern diffusion models. In offline reinforcement learning, SYNTHER allows training from extremely small datasets, scaling up policy and value networks, and high levels of data compression. In online reinforcement learning, the additional data allows agents to use much higher update-to-data ratios leading to increased sample efficiency.

We have demonstrated that SYNTHER is a scalable approach and believe that extending it to more settings would unlock extremely exciting new capabilities for RL agents. SYNTHER could readily be extended to n-step formulations of experience replay by simply expanding the input space of the diffusion model. Furthermore, whilst we demonstrated an effective method to generate synthetic data in latent space for pixel-based settings, exciting future work could involve generating transitions in the original image space. In particular, one could consider fine-tuning large pre-trained foundation models [59] and leveraging their generalization capability to synthesize novel views and configurations of a pixel-based environment. Finally, by using guidance for diffusion models [29], the generated synthetic data could be biased towards certain modes, resulting in transferable and composable sampling strategies for RL algorithms.

Acknowledgments

Cong Lu is funded by the Engineering and Physical Sciences Research Council (EPSRC). Philip Ball is funded through the Willowgrove Studentship. The authors would like to thank the anonymous Reincarnating Reinforcement Learning Workshop at ICLR 2023 and NeurIPS 2023 reviewers for positive and constructive feedback which helped to improve the paper. We would also like to thank Shimon Whiteson, Jakob Foerster, Tim Rocktäschel and Ondrej Bajgar for reviewing earlier versions of this work.

References

[1] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei Zhang. Human-timescale adaptation in an open-ended task space, 2023. URL https://arxiv.org/abs/2301.07608.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

[3] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

[4] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[5] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7436–7447. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/3d3d286a8d153a4a58156d0e02d8570c-Paper.pdf.

[6] Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

[7] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J. Fleet. Synthetic data from diffusion models improves imagenet classification, 2023.

[8] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 619–629. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ball21a.html.

[9] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[12] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

[13] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation, 2023. URL https://arxiv.org/abs/2302.06671.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] 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*, 2018.

[16] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OpC-9aBBVJe.

[17] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation, 2023. URL https://arxiv.org/abs/2302.00111.

[18] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[19] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[20] E. C. Fieller, H. O. Hartley, and E. S. Pearson. Tests for rank correlation coefficients. i. *Biometrika*, 44(3/4):470–481, 1957. ISSN 00063444. URL http://www.jstor.org/stable/2332878.

[21] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[22] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[26] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition?, 2022. URL https://arxiv.org/abs/2210.07574.

[27] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations, 2017. URL https://arxiv.org/abs/1704.03732.

[28] Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning, 2023. URL https://arxiv.org/abs/2301.13442.

[29] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[31] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=6nbpPqUCIi7.

[32] Vincent Huang, Tobias Ley, Martha Vlachou-Konchylaki, and Wenfeng Hu. Enhanced experience replay generation for efficient reinforcement learning, 2017. URL https://arxiv.org/abs/1705.08245.

[33] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

[34] Baris Imre. An investigation of generative replay in deep reinforcement learning, January 2021. URL http://essay.utwente.nl/85772/.

[35] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

[36] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[37] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[38] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=k7FuTOWMOc7.

[39] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f7efa4f864ae9b88d43527f4b14f750f-Paper.pdf.

[40] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[41] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

[42] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models, 2022.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, 2012.

[44] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

[45] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf.

[46] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[47] Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *Advances in neural information processing systems*, 33:12861–12872, 2020.

[48] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating statistical overfitting. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=14-kr46GvP-.

[49] Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J. Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=zz9hXVhf40.

[50] Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations, 2022. URL https://arxiv.org/abs/2206.04779.

[51] A.C.P.P. Ludjen. Generative replay in deep reinforcement learning, June 2021. URL http://essay.utwente.nl/87315/.

[52] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015. doi: 10.1038/nature14236.

[54] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016. doi: 10.1109/DSAA.2016.49.

[55] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Pv1GPQzRrC8.

[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[57] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

[58] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[59] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[60] Mert Bulent Sariyildiz, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[61] Stefan Schaal. Learning from demonstration. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf.

[62] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

[63] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

[64] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WVX0NNVBBkV.

[65] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf.

[66] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[67] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[68] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

[69] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[70] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP.

[71] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

[72] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[73] 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 I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[74] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

[75] Andrew Wagenmaker and Aldo Pacchiano. Leveraging offline data in online reinforcement learning, 2022. URL https://arxiv.org/abs/2211.04974.

[76] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

[77] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=_SJ-_yyes8.

[78] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf.

[79] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Brian Ichter, Karol Hausman, and Fei Xia. Scaling robot learning with semantically imagined experience, 2023. URL https://arxiv.org/abs/2302.11550.

Limitations and Future Work

Our work shows the remarkable efficacy of diffusion models as a method for data augmentation in reinforcement learning—we believe we have barely scratched the surface of what is possible in this space. One opportunity for future work is leveraging the strength of diffusion models at generating conditional data, i.e. data targeted towards a particular label or statistic, whereas our models are entirely unconditional. We could imagine conditioning the model to generate higher TD-error data akin to Prioritized Experience Replay (Schaul et al., 2016) to further accelerate training. Along the same lines, we could also explicitly encourage the diffusion model to generate more diverse data (Azizi et al., 2023) or more on-policy data (Jackson et al., 2023; Rigter et al., 2023). When we are in the multi-task or meta-RL (Zintgraf et al., 2021a,b) setting, we could also consider generating data for out-of-distribution tasks for better generalization.

Next, in our online experiments, the diffusion model took a more "passive" role, simply consuming and producing data. We note that there is a rich body of work in *active learning* with forward dynamics models (Ball et al., 2020), where samples are collected to improve the model with as few samples as possible. One method of doing so is to target model uncertainty—diffusion models can be used for uncertainty estimation by evaluating the likelihood of a sample (Karras et al., 2022) or by measuring the empirical distribution over many samples (Han et al., 2022).

Finally, for our pixel-based experiments, whilst producing data in the latent space of a fixed pre-trained encoder allows us to use a fast MLP for diffusion, a fixed encoder also limits the maximum performance. One challenge of generating transitions in the original image space is that the transitions are naturally multi-modal. This is because we must generate a high-dimensional image alongside a low-dimensional action and reward simultaneously. Appropriately handling multi-modality is an open research question—while some progress has been made in multi-modal diffusion (Ruan et al., 2023; Tang et al., 2023) for combinations of text, video, and audio, our specific application will likely need careful design. Related to this is the advent of pre-trained

generative foundation models (Hu et al., 2023; Rombach et al., 2022) which are trained on internet-level quantities of images and video. Fine-tuning these models could greatly reduce the requirements for our algorithm, and also aid generalization by leveraging pre-existing concepts.

Part II

Reinforcement Learning from Offline Data

5

On Pathologies in KL-Regularized Reinforcement Learning from Expert Demonstrations.

We begin the reinforcement learning from offline data section by investigating methods to accelerate online reinforcement learning with expert demonstrations. As discussed in Section 2.3.3, a popular algorithm for this setting is KL-regularized RL (Galashov et al., 2019; Rawlik et al., 2012; Schulman et al., 2017a; Todorov, 2007) where we optimize the following augmented objective

$$\tilde{J}(\pi) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) - \alpha \mathrm{KL}(\pi(\cdot|s_t) \parallel \pi_0(\cdot|s_t))\right)\right] \tag{5.1}$$

given a base policy $\pi_0 : \mathcal{S} \to \mathcal{A}$ learned from a set of expert demonstrations *without reward* via supervised behavioral cloning, and regularization strength α. The form of the KL divergence in Equation (5.1) is the "reverse" or mode-seeking variant (Chan et al., 2022), as opposed to "forward" or mode-covering if the roles of π and π_0 are reversed. One reason for choosing the reverse KL is that in the event of multiple expert action modes, it is often better to choose one mode rather than averaging over actions which would likely be suboptimal.

In typical actor-critic settings (Haarnoja et al., 2018), both the online policy and behavioral prior will be realized as parameterized Gaussians; i.e. $\pi\left(\cdot|s_t\right) = \mathcal{N}\left(\mu\left(s_t\right), \sigma\left(s_t\right)\right)$ and $\pi_0\left(\cdot|s_t\right) = \mathcal{N}\left(\mu_0\left(s_t\right), \sigma_0\left(s_t\right)\right)$. This allows us to compute the KL divergence exactly which has the form

$$\mathrm{KL}(\pi(\cdot|s_t) \parallel \pi_0(\cdot|s_t)) \propto \log \frac{\sigma_0(s_t)}{\sigma(s_t)} + \frac{\sigma^2(s_t) + (\mu(s_t) - \mu_0(s_t))^2}{2\sigma_0^2(s_t)} \tag{5.2}$$

Thus, the divergence in mean predictions between the online policy and the prior, $(\mu(s_t) - \mu_0(s_t))^2$, is approximately scaled by the predictive variance of the base policy, $\sigma_0^2(s_t)$. If the predictive variances were constant here, the KL-divergence would reduce to the standard BC loss which has appeared in other works (Fujimoto and Gu, 2021; Goecks et al., 2019).

This scaling allows the KL-regularized term to improve upon the standard BC loss by encouraging exploration and reducing regularization strength where the variance of π_0 is high and the opposite when low. As a consequence, the behavioral policy should appropriately quantify uncertainty—being certain when close to the expert data and uncertain when out-of-distribution. However, herein lies a pathology when naïve choices for π_0 are made. We show empirically that commonly chosen behavioral policy classes suffer from extremely high confidence (or uncertainty collapse) in out-of-distribution states where their predictions are more likely to be poor. This holds for all parametric uncertainty quantification methods that we evaluate including deep ensembles (Lakshminarayanan et al., 2017) and Bayesian neural networks with Monte Carlo dropout (Gal and Ghahramani, 2016). Furthermore, we prove theoretically that this type of uncertainty collapse leads to exploding gradients during online RL training and instability. Whilst it is possible to side-step this issue by exactly matching the behavioral prior, this precludes any further improvement from online RL.

We resolve this pathology by instead turning to non-parametric behavioral reference policies using Gaussian processes (GPs) as discussed in Section 2.1.5; this forms the backbone of our algorithm *Non-Parametric Prior Actor–Critic* (N-PPAC). GPs

allow us to perform exact Bayesian inference over modestly sized datasets, which is ideal for our use case, as expert demonstration sets are typically small in number. In contrast to parametric behavioral priors, GPs provide well-calibrated predictive uncertainty estimates which we verify. This enables KL-regularized reinforcement learning to significantly outperform prior state-of-the-art approaches on a variety of challenging locomotion and dexterous hand manipulation tasks.

Tim G. J. Rudner*, **Cong Lu***, Michael A. Osborne, Yarin Gal, and Yee Whye Teh. On Pathologies in KL-Regularized Reinforcement Learning from Expert Demonstrations. **In NeurIPS, 2021.**

On Pathologies in KL-Regularized Reinforcement Learning from Expert Demonstrations

Tim G. J. Rudner[*,†] University of Oxford

Cong Lu[*] University of Oxford

Michael A. Osborne University of Oxford

Yarin Gal University of Oxford

Yee Whye Teh University of Oxford

Abstract

KL-regularized reinforcement learning from expert demonstrations has proved successful in improving the sample efficiency of deep reinforcement learning algorithms, allowing them to be applied to challenging physical real-world tasks. However, we show that KL-regularized reinforcement learning with behavioral reference policies derived from expert demonstrations can suffer from pathological training dynamics that can lead to slow, unstable, and suboptimal online learning. We show empirically that the pathology occurs for commonly chosen behavioral policy classes and demonstrate its impact on sample efficiency and online policy performance. Finally, we show that the pathology can be remedied by *non-parametric* behavioral reference policies and that this allows KL-regularized reinforcement learning to significantly outperform state-of-the-art approaches on a variety of challenging locomotion and dexterous hand manipulation tasks.

1 Introduction

Reinforcement learning (RL) [15, 24, 46, 47] is a powerful paradigm for learning complex behaviors. Unfortunately, many modern reinforcement learning algorithms require agents to carry out millions of interactions with their environment to learn desirable behaviors, making them of limited use for a wide range of practical applications that cannot be simulated [8, 28]. This limitation has motivated the study of algorithms that can incorporate pre-collected offline data into the training process, either fully offline or with online exploration, to improve sample efficiency, performance, and reliability [2, 6, 16, 23, 52, 53]. An important and well-motivated subset of these methods consists of approaches for efficiently incorporating expert demonstrations into the learning process [5, 11, 18, 42].

Reinforcement learning with Kullback-Leibler (KL) regularization is a particularly successful approach for doing so [3, 27, 29, 31, 44, 51]. In KL-regularized reinforcement learning, the standard reinforcement learning objective is augmented by a Kullback-Leibler divergence term that penalizes dissimilarity between the online policy and a behavioral reference policy derived from expert demonstrations. The resulting regularized objective pulls the agent's online policy towards the behavioral reference policy while also allowing it to improve upon the behavioral reference policy by exploring and interacting with the environment. Recent advances that leverage explicit or implicit KL-regularized objectives, such as BRAC [51], ABM [44], and AWAC [27], have shown that KL-regularized reinforcement learning from expert demonstrations is able to significantly improve the sample efficiency of online training and reliably solve challenging environments previously unsolved by standard deep reinforcement learning algorithms.

[*]Equal contribution. [†] Corresponding author: tim.rudner@cs.ox.ac.uk.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

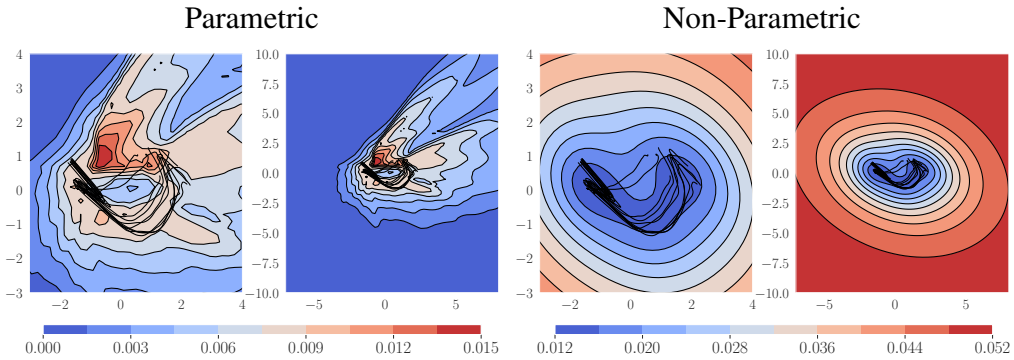


Figure 1: Predictive variances of non-parametric and parametric behavioral policies on a low dimensional representation (the first two principal components) of a 39-dimensional dexterous hand manipulation state space (see "door-binary-v0" in Figure 5). **Left**: Parametric neural network Gaussian behavioral policy $\pi_\psi(\cdot \mid \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{s}), \boldsymbol{\sigma}^2_\psi(\mathbf{s}))$. **Right**: Non-parametric Gaussian process posterior behavioral policy $\pi_{\mathcal{GP}}(\cdot \mid \mathbf{s}, \mathcal{D}_0) = \mathcal{GP}(\boldsymbol{\mu}_0(\mathbf{s}), \boldsymbol{\Sigma}_0(\mathbf{s}, \mathbf{s}'))$. Expert trajectories $\mathcal{D}$ used to train the behavioral policies are shown in black. The GP predictive variance is well-calibrated: It is small near the expert trajectories and large in other parts of the state space. In contrast, the neural network predictive variance is poorly calibrated: It is relatively small on the expert trajectories, and collapses to near zero elsewhere. Note the significant difference in scales.

Contributions. In this paper, we show that despite some empirical success, KL-regularized reinforcement learning from expert demonstrations can suffer from previously unrecognized pathologies that lead to instability and sub-optimality in online learning. To summarize, our core contributions are as follows:

- We illustrate empirically that commonly used classes of parametric behavioral policies experience a collapse in predictive variance about states away from the expert demonstrations.
- We demonstrate theoretically and empirically that KL-regularized reinforcement learning algorithms can suffer from pathological training dynamics in online learning when regularized against behavioral policies that exhibit such a collapse in predictive variance.
- We show that the pathology can be remedied by *non-parametric* behavioral policies, whose predictive variances are well-calibrated and guaranteed not to collapse about previously unseen states, and that fixing the pathology results in online policies that significantly outperform state-of-the-art approaches on a range of challenging locomotion and dexterous hand manipulation tasks.

The left panel of Figure 1 shows an example of the collapse in predictive variance away from the expert trajectories in parametric behavioral policies. In contrast, the right panel of Figure 1 shows the predictive variance of a non-parametric behavioral policy, which—unlike in the case of the parametric policy—increases off the expert trajectories. By avoiding the pathology, we obtain a stable and reliable approach to sample-efficient reinforcement learning, applicable to a wide range of reinforcement learning algorithms that leverage KL-regularized objectives.[2]

2 Background

We consider the standard reinforcement learning setting where an agent interacts with a discounted Markov Decision Process (MDP) [46] given by a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$ are the transition dynamics, $r(\mathbf{s}_t, \mathbf{a}_t)$ is the reward function, and γ is a discount factor. $\rho_\pi(\boldsymbol{\tau}_t)$ denotes the state–action trajectory distribution from time t induced by a policy $\pi(\cdot \mid \mathbf{s}_t)$. The discounted return from time step t is given by $R(\boldsymbol{\tau}_t) = \sum_{k=t}^{\infty} \gamma^k r(\mathbf{s}_k, \mathbf{a}_k)$ for $t \in \mathbb{N}_0$. The standard reinforcement learning objective to be maximized is the expected discounted return $J_\pi(\boldsymbol{\tau}_0) = \mathbb{E}_{\rho_\pi(\boldsymbol{\tau}_0)}[R(\boldsymbol{\tau}_0)]$ under the policy trajectory distribution.

2.1 Improving and Accelerating Online Training via Behavioral Cloning

We consider settings where we have a set of expert demonstrations *without reward*, $\mathcal{D}_0 = \{(\mathbf{s}_n, \mathbf{a}_n)\}_{n=1}^N = \{\bar{\mathbf{S}}, \bar{\mathbf{A}}\}$, which we would like to use to speed up and improve online learn-

[2]Code and visualizations of our results can be found at https://sites.google.com/view/nppac.

ing [5, 42]. A standard approach for turning expert trajectories into a policy is behavioral cloning [1, 4] which involves learning a mapping from states in the expert demonstrations to their corresponding actions, that is, $\pi_0 : \mathcal{S} \to \mathcal{A}$. As such, behavioral cloning does not assume or require access to a reward function and only involves learning a mapping from states to action in a supervised fashion.

Since expert demonstrations are costly to obtain and often only available in small number, behavioral cloning alone is typically insufficient for agents to learn good policies in complex environments and has to be complemented by a method that enables the learner to build on the cloned behavior by interacting with the environment. A particularly successful and popular class of algorithms used for incorporating behavioral policies into online training is KL-regularized reinforcement learning [10, 37, 43, 48].

2.2 KL-Regularized Objectives in Reinforcement Learning

KL-regularized reinforcement learning modifies the standard reinforcement learning objective by augmenting the return with a negative KL divergence term from the learned policy π to a reference policy π_0, given a temperature parameter α. The resulting discounted return from time step $t \in \mathbb{N}_0$ is then given by

$$\tilde{R}(\boldsymbol{\tau}_t) = \sum_{k=t}^{\infty} \gamma^k \big[r(\mathbf{s}_k, \mathbf{a}_k) - \alpha \mathbb{D}_{\mathrm{KL}}(\pi(\cdot \,|\, \mathbf{s}_k) \parallel \pi_0(\cdot \,|\, \mathbf{s}_k)) \big] \tag{1}$$

and the reinforcement learning objective becomes $\tilde{J}_\pi(\boldsymbol{\tau}_0) = \mathbb{E}_{\rho_\pi(\boldsymbol{\tau}_0)}[\tilde{R}(\boldsymbol{\tau}_0)]$. When the reference policy π_0 is given by a uniform distribution, we recover the entropy-regularized reinforcement learning objective used in Soft Actor–Critic (SAC) [13] up to an additive constant.

Under a uniform reference policy π_0, the resulting objective encourages exploration, while also choosing high-reward actions. In contrast, when π_0 is non-uniform, the agent is discouraged to explore areas of the state space $\mathcal{S}$ where the variance of $\pi_0(\cdot \,|\, \mathbf{s})$ is low (i.e., more certain) and encouraged to explore areas of the state space where the variance of $\pi_0(\cdot \,|\, \mathbf{s})$ is high. The KL-regularized reinforcement learning objective can be optimized via policy–gradient and actor–critic algorithms.

2.3 KL-Regularized Actor–Critic

An optimal policy π that maximizes the expected KL-augmented discounted return $\tilde{J}_\pi$ can be learned by directly optimizing the policy gradient $\nabla_\pi \tilde{J}_\pi$. However, this policy gradient estimator exhibits high variance, which can lead to unstable learning. Actor–critic algorithms [7, 17, 32, 38] attempt to reduce this variance by making use of the state value function $V^\pi(\mathbf{s}_t) = \mathbb{E}_{\rho_\pi(\boldsymbol{\tau}_t)}[\tilde{R}(\boldsymbol{\tau}_t) \,|\, \mathbf{s}_t]$ or the state–action value function $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\rho_\pi(\boldsymbol{\tau}_t)}[\tilde{R}(\boldsymbol{\tau}_t) \,|\, \mathbf{s}_t, \mathbf{a}_t]$ to stabilize training.

Given a reference policy $\pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t)$, the state value function can be shown to satisfy the modified Bellman equation

$$V^\pi(\mathbf{s}_t) \doteq \mathbb{E}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)] - \alpha \mathbb{D}_{\mathrm{KL}}\big(\pi(\cdot \,|\, \mathbf{s}_t) \,||\, \pi_0(\cdot \,|\, \mathbf{s}_t)\big)$$

with a recursively defined Q-function

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \doteq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)}[V^\pi(\mathbf{s}_{t+1})].$$

Instead of directly optimizing the objective function $\tilde{J}_\pi$ via the policy gradient, actor–critic methods alternate between policy evaluation and policy improvement [7, 13]:

Policy Evaluation. During the policy evaluation step, $Q^\pi_\theta(\mathbf{s}, \mathbf{a})$, parameterized by parameters θ, is trained by minimizing the Bellman residual

$$J_Q(\theta) \doteq \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \Big[\big(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)}[V_{\bar{\theta}}(\mathbf{s}_{t+1})]) \big)^2 \Big], \tag{2}$$

where $\mathcal{D}$ is a replay buffer and $\bar{\theta}$ is a stabilizing moving average of parameters.

Policy Improvement. In the policy improvement step, the policy π_ϕ, parameterized by parameters ϕ, is updated towards the exponential of the KL-augmented Q-function,

$$J_\pi(\phi) \doteq \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\alpha \mathbb{D}_{\mathrm{KL}}(\pi_\phi(\cdot \,|\, \mathbf{s}_t) \parallel \pi_0(\cdot \,|\, \mathbf{s}_t)) \right] - \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi(\cdot|\mathbf{s}_t)} \left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \right] \right], \tag{3}$$

with states sampled from a replay buffer $\mathcal{D}$ and actions sampled from the parameterized online policy π_ϕ. The following sections will focus on the policy improvement objective and how certain types of references policies can lead to pathologies when optimizing $J_\pi(\phi)$ with respect to ϕ.

3 Identifying the Pathology

In this section, we investigate the effect of KL-regularization on the training dynamics. To do so, we first consider the properties of the KL divergence to identify a potential failure mode for KL-regularized reinforcement learning. Next, we consider parametric Gaussian behavioral reference policies commonly used in practice for continuous control tasks [13, 51] and show that for Gaussian behavioral reference policies with small predictive variance, the policy improvement objective suffers from exploding gradients with respect to the policy parameters ϕ. We confirm that this failure occurs empirically and demonstrate that it results in slow, unstable, and suboptimal online learning. Lastly, we show that various regularization techniques used for estimating behavioral policies are unable to prevent this failure and also lead to suboptimal online policies.

3.1 When Are KL-Regularized Reinforcement Learning Objectives Meaningful?

We start by considering the properties of the KL divergence and discuss how these properties can lead to potential failure modes in KL-regularized objectives. A well-known property of KL-regularized objectives in the variational inference literature is the occurrence of singularities when the support of one distribution is not contained in the support of the other.

To illustrate this problem, we consider the case of Gaussian behavioral and online policies commonly used in practice. Mathematically, the KL divergence between two full Gaussian distributions is always finite and well-defined. Hence, we might hope KL-regularized reinforcement learning with Gaussian behavioral and online policies to be unaffected by the failure mode described above. However, the support of a Gaussian online policy $\pi_\phi(\cdot \mid \mathbf{s}_t)$ will not be contained in the support of a behavioral reference policy $\pi_0(\cdot \mid \mathbf{s}_t)$ as the predictive variance $\boldsymbol{\sigma}_0^2(\mathbf{s}_t)$ tends to zero, and hence $\mathbb{D}_{\text{KL}}(\pi_\phi(\cdot \mid \mathbf{s}_t) \parallel \pi_0(\cdot \mid \mathbf{s}_t)) \to \infty$ as $\boldsymbol{\sigma}_0^2(\mathbf{s}_t) \to 0$. In other words, as the variance of a behavioral reference policy tends to zero and the behavioral distribution becomes degenerate, the KL divergence blows up to infinity [25]. While in practice, Gaussian behavioral policy would not operate in the limit of zero variance, the functional form of the KL divergence between (univariate) Gaussians,

$$\mathbb{D}_{\text{KL}}(\pi_\phi(\cdot \mid \mathbf{s}_t) \parallel \pi_0(\cdot \mid \mathbf{s}_t)) \propto \log \frac{\boldsymbol{\sigma}_0(\mathbf{s}_t)}{\boldsymbol{\sigma}_\phi(\mathbf{s}_t)} + \frac{\boldsymbol{\sigma}_\phi^2(\mathbf{s}_t) + (\boldsymbol{\mu}_\phi(\mathbf{s}_t) - \boldsymbol{\mu}_0(\mathbf{s}_t))^2}{2\boldsymbol{\sigma}_0^2(\mathbf{s}_t)},$$

implies a continuous, quadratic increase in the magnitude of the divergence as $\boldsymbol{\sigma}_0(\mathbf{s}_t)$ decreases, further exacerbated by a large difference in predictive means, $|\boldsymbol{\mu}_\phi(\mathbf{s}_t) - \boldsymbol{\mu}_0(\mathbf{s}_t)|$.

As a result, for Gaussian behavioral reference policies $\pi_0(\cdot \mid \mathbf{s}_t)$ that assign very low probability to sets of points in sample space far away from the distribution's mean $\boldsymbol{\mu}_0(\mathbf{s}_t)$, computing the KL divergence can result in divergence values so large to cause numerical instabilities and arithmetic overflow. Hence, even for a suitably chosen behavioral reference policy class, vanishingly small behavioral reference policy predictive variances can cause the KL divergence to 'blow up' and cause numerical issues at evaluation points far away from states in the expert demonstrations.

One way to address this failure mode may be to lower-bound the output of the variance network (e.g., by adding a small constant bias). However, placing a floor on the predictive variance of the behavioral reference policy is not sufficient to encourage effective learning. While it would prevent the KL divergence from blowing up, it would also lead to poor gradient signals, as well-calibrated predictive variance estimates that *increase* on states far away from the expert trajectories are necessary to keep the KL penalty from pulling the predictive mean of the online policy towards poor behavioral reference policy predictive means on states off the expert trajectories. Another possible solution could be to use heavy-tailed behavioral reference policies distributions, for example, Laplace distributions, to avoid pathological training dynamics. However, in Appendix B.3 we show that Laplace behavioral reference policies also suffer from pathological training dynamics, albeit less severely.

In the following sections, we explain how an explosion in $\mathbb{D}_{\text{KL}}(\pi_\phi(\cdot \mid \mathbf{s}_t) \parallel \pi_0(\cdot \mid \mathbf{s}_t))$ caused by small $\boldsymbol{\sigma}_0^2(\mathbf{s}_t)$ affects the gradients of $J_\pi(\phi)$ in KL-regularized RL and discuss of how and why $\boldsymbol{\sigma}_0^2(\mathbf{s}_t)$ may tend to zero in practice.

3.2 Exploding Gradients in KL-Regularized Reinforcement Learning Objectives

To understand how small predictive variances in behavioral reference policies can affect—and possibly destabilize—online training in KL-regularized RL, we consider the contribution of the behavioral

reference policy's variance to the gradient of the policy objective in Equation (3). Compared to entropy-regularized actor–critic methods (SAC, Haarnoja et al. [13]), which implicitly regularize against a uniform policy, the gradient estimator $\hat{\nabla}_\phi J_\pi(\phi)$ in KL-regularized RL gains an extra scaling term $\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \mid \mathbf{s}_t)$, the gradient of the prior log-density evaluated actions $\mathbf{a}_t \sim \pi_\phi(\cdot \mid \mathbf{s})$:

Proposition 1 (Exploding Gradients in KL-Regularized RL)**.** *Let $\pi_0(\cdot \mid \mathbf{s})$ be a Gaussian behavioral reference policy with mean $\boldsymbol{\mu}_0(\mathbf{s}_t)$ and variance $\boldsymbol{\sigma}_0^2(\mathbf{s}_t)$, and let $\pi_\phi(\cdot \mid \mathbf{s})$ be an online policy with reparameterization $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$ and random vector ϵ_t. The gradient of the policy loss with respect to the online policy's parameters ϕ is then given by*

$$\begin{aligned}\hat{\nabla}_\phi J_\pi(\phi) = \big(&\alpha \nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t) - \alpha \nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \mid \mathbf{s}_t) \\ &- \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)\big) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) + \alpha \nabla_\phi \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)\end{aligned} \tag{4}$$

with $\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \mid \mathbf{s}_t) = -\frac{\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)}{\boldsymbol{\sigma}_0^2(\mathbf{s}_t)}$. *For fixed* $|\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)|$, $\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t|\mathbf{s}_t)$ *grows as* $\mathcal{O}(\boldsymbol{\sigma}_0^{-2}(\mathbf{s}_t))$*; thus,*

$$|\,\hat{\nabla}_\phi J_\pi(\phi)\,| \to \infty \quad \textit{as} \quad \boldsymbol{\sigma}_0^2(\mathbf{s}_t) \to 0 \quad \textit{whenever} \quad \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \neq 0.$$

Proof. See Appendix A.1. □

This result formalizes the intuition presented in Section 3.1 that a behavioral reference policy with a sufficiently small predictive variance may cause KL-regularized reinforcement learning to suffer from pathological training dynamics in gradient-based optimization. The smaller the behavioral reference policy's predictive variance, the more sensitive the policy objective's gradients will be to differences in the means of the online and behavioral reference policies. As a result, for behavioral reference policies with small predictive variance, the KL divergence will heavily penalize online policies whose predictive means diverge from the predictive means of the behavioral policy—even in regions of the state space away from the expert trajectory where the behavioral policy's mean prediction is poor.

3.3 Predictive Uncertainty Collapse Under Parametric Policies

The most commonly used method for estimating behavioral policies is maximum likelihood estimation (MLE) [44, 51], where we seek $\pi_0 \doteq \pi_{\psi^\star}$ with $\psi^\star \doteq \arg\max_\psi \left\{\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}_0}[\log \pi_\psi(\mathbf{a} \mid \mathbf{s})]\right\}$ for a parametric behavioral policy π_ψ. In practice, π_ψ is often assumed to be Gaussian, $\pi_\psi(\cdot \mid \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{s}), \boldsymbol{\sigma}_\psi^2(\mathbf{s}))$, with $\boldsymbol{\mu}_\psi(\mathbf{s})$ and $\boldsymbol{\sigma}_\psi^2(\mathbf{s})$ parameterized by a neural network.

While maximizing the likelihood of the expert trajectories under the behavioral policy is a sensible choice for behavioral cloning, the limited capacity of the neural network parameterization can produce unwanted behaviors in the resulting policy. The maximum likelihood objective ensures that the behavioral policy's predictive mean reflects the expert's actions and the predictive variance the (aleatoric) uncertainty inherent in the expert trajectories.

However, the maximum likelihood objective encourages parametric policies to use their model capacity toward fitting the expert demonstrations and reflecting the aleatoric uncertainty in the data. As a result, for states off the expert trajectories, the policy can become degenerate and collapse to point predictions instead of providing meaningful predictive variance estimates that reflect that the behavioral policy ought to be highly uncertain about its predictions in previously unseen regions of the state space. Similar behaviors are well-known in parametric probabilistic models and well-documented in the approximate Bayesian inference literature [33, 39].

Figure 1 demonstrates the collapse in predictive variance under maximum likelihood estimation in a low-dimensional representation of the "door-binary-v0" dexterous hand manipulation environment. It shows that while the predictive variance is small close to the expert trajectories (depicted as black lines), it rapidly decreases further away from them. Examples of variance collapse in other environments are presented in Appendix B.6. Figure 2 shows that the predictive variance off the expert trajectories consistently decreases during training. As shown in Proposition 1, such a collapse in predictive variance can

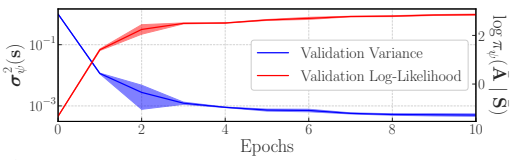


Figure 2: Collapse in the predictive variance (in **blue**) of a Gaussian behavioral policy parameterized by a neural network when training via maximum likelihood estimation. Lines and shaded regions denote means and standard deviations over five random seeds, respectively.

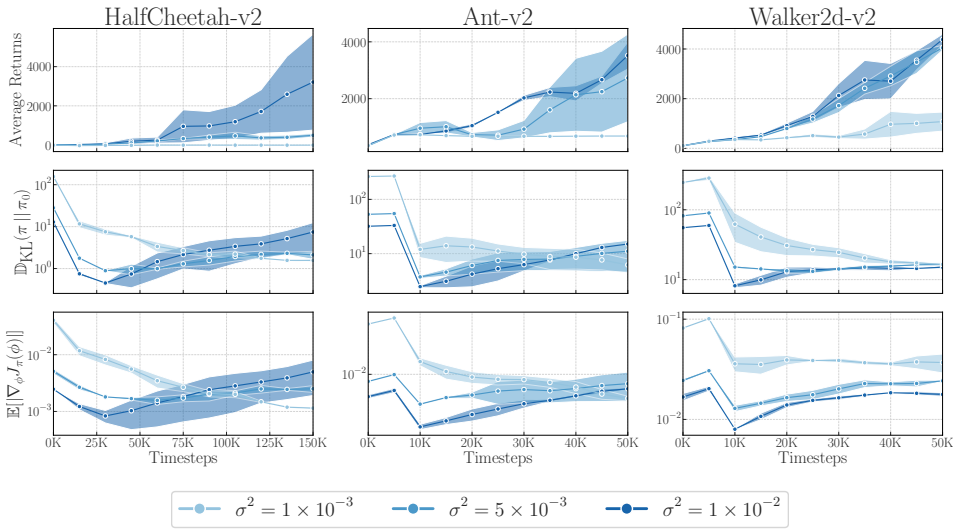


Figure 3: Ablation study showing the effect of predictive variance collapse on the performance of KL-regularized RL on MuJoCo environments. The plots show the average return of the learned policy, the magnitude of the KL penalty, and the magnitude of the average absolute gradients of the policy loss during online training. The lighter the shading, the lower the behavioral policy's predictive variance.

result in pathological training dynamics in KL-regularized online learning—steering the online policy towards suboptimal trajectories in regions of the state space far away from the expert demonstrations and deteriorating performance.

Effect of regularization on uncertainty collapse. To prevent a collapse in the behavioral policy's predictive variance, prior work proposed adding entropy or Tikhonov regularization to the MLE objective [51]. However, doing so does not succeed in preventing a collapse in predictive variance off the expert demonstration trajectories, as we show in Appendix A.3. Deep ensembles [20], whose predictive mean and variance are computed from the predictive means and variances of multiple Gaussian neural networks, are a widely used method for uncertainty quantification in regression settings. However, model ensembling can be costly and unreliable, as it requires training multiple neural networks from scratch and does not guarantee well-calibrated uncertainty estimates [39, 49]. We provide visualizations in Appendix B.5 which show that ensembling multiple neural network policies does not fully prevent a collapse in predictive variance.

3.4 Empirical Confirmation of Uncertainty Collapse

To confirm Proposition 1 empirically and assess the effect of the collapse in predictive variance on the performance of KL-regularized RL, we perform an ablation study where we fix the predictive mean function of a behavioral policy to a mean function that attains 60% of the optimal performance and vary the magnitude of the policy's predictive variance. Specifically, we set the behavioral policy's predictive variance to different constant values in the set $\{1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}\}$ (following a similar implementation in Nair et al. [27]).[3] The results of this experiment are shown in Figure 3, which shows the average returns, the KL divergence, and the average absolute gradients of the policy loss over training. The plots confirm that as the predictive variance of the offline behavioral policy tends to zero, the KL terms and average policy gradient magnitude explode as implied by Proposition 1, leading to unstable training and a collapse or dampening in average returns.

In other words, even for behavioral policies with accurate predictive means, smaller predictive variances slow down or even entirely prevent learning good behavioral policies. This observation confirms that the pathology identified in Proposition 1 occurs in practice and that it can have a significant impact on KL-regularized RL from expert demonstrations, calling into question the usefulness of KL regularization as a means for accelerating and improving online training. In Appendix B.1, we show that an analogous relationship exists for the gradients of the Q-function loss.

[3] We attempted to use smaller values, but the gradients grew too large and caused arithmetic overflow.

4 Fixing the Pathology

In order to address the collapse in predictive uncertainty for behavioral policies parameterized by a neural network trained via MLE, we specify a *non-parametric* behavioral policy whose predictive variance is *guaranteed* not to collapse about previously unseen states. Noting that KL-regularized RL with a behavioral policy can be viewed as approximate Bayesian inference with an empirical prior policy [13, 21, 40], we propose *Non-Parametric Prior Actor–Critic* (N-PPAC), an off-policy temporal difference algorithm for improved, accelerated, and stable online learning with behavioral policies.

4.1 Non-Parametric Gaussian Processes Behavioral Policies

Gaussian processes (GPs) [36] are models over functions defined by a mean $m(\cdot)$ and covariance function $k(\cdot, \cdot)$. When defined in terms of a non-parametric covariance function, that is, a covariance function constructed from infinitely many basis functions, we obtain a non-degenerate GP, which has sufficient capacity to prevent a collapse in predictive uncertainty away from the training data. Unlike parametric models, whose capacity is limited by their parameterization, a non-parametric model's capacity *increases* with the amount of training data.

Considering a non-parametric GP behavioral policy, $\pi_0(\cdot \,|\, \mathbf{s})$, with

$$\mathbf{A} \,|\, \mathbf{s} \sim \pi_0(\cdot \,|\, \mathbf{s}) = \mathcal{GP}\big(m(\mathbf{s}), k(\mathbf{s}, \mathbf{s}')\big), \tag{5}$$

we can obtain a *non-degenerate* posterior distribution over actions conditioned on the offline data $\mathcal{D}_0 = \{\bar{\mathbf{S}}, \bar{\mathbf{A}}\}$ with actions sampled according to the

$$\mathbf{A} \,|\, \mathbf{s}, \mathcal{D}_0 \sim \pi_0(\cdot \,|\, \mathbf{s}, \mathcal{D}_0) = \mathcal{GP}\big(\boldsymbol{\mu}_0(\mathbf{s}), \boldsymbol{\Sigma}_0(\mathbf{s}, \mathbf{s}')\big), \tag{6}$$

with

$$\mu(\mathbf{s}) = m(\mathbf{s}) + k(\mathbf{s}, \bar{\mathbf{S}}) k(\bar{\mathbf{S}}, \bar{\mathbf{S}})^{-1} (\bar{\mathbf{A}} - m(\bar{\mathbf{A}})) \quad \text{and} \quad \Sigma(\mathbf{s}, \mathbf{s}') = k(\mathbf{s}, \mathbf{s}') + k(\mathbf{s}, \bar{\mathbf{S}}) k(\bar{\mathbf{S}}, \bar{\mathbf{S}})^{-1} k(\bar{\mathbf{S}}, \mathbf{s}').$$

To obtain this posterior distribution, we perform exact Bayesian inference, which naively scales as $\mathcal{O}(N^3)$ in the number of training points N, but Wang et al. [50] show that exact inference in GP regression can be scaled to $N > 1,000,000$. Since expert demonstrations usually contain less than 100k datapoints, non-parametric GP behavioral policies are applicable to a wide array of real-world tasks. For an empirical evaluation of the time complexity of using a GP prior, see Section 5.5.

Figure 1 confirms that the non-parametric GP's predictive variance is well-calibrated: It is small in magnitude in regions of the state space near the expert trajectories and large in magnitude in other regions of the state space. While actor–critic algorithms like SAC implicitly use a uniform prior to explore the state space, using a behavioral policy with a well-calibrated predictive variance has the benefit that in regions of the state space close to the expert demonstrations the online policy learns to match the expert, while elsewhere the predictive variance increases and encourages exploration.

Algorithmic details. In our experiments, we use a KL-regularized objective with a standard actor–critic implementation and Double DQN [14]. Pseudocode is provided in (Appendix C.1).

5 Empirical Evaluation

We carry out a comparative empirical evaluation of our proposed approach vis-à-vis related methods that integrate offline data into online training. We provide a detailed description of the algorithms we compare against in Appendix A.4. We perform experiments on the MuJoCo benchmark suite and the substantially more challenging dexterous hand manipulation suite with sparse rewards.

We show that KL-regularized RL with a non-parametric behavioral reference policy can rapidly learn to solve difficult high-dimensional continuous control problems given only a small set of expert demonstrations and (often significantly) outperforms state-of-the-art methods, including ones that use offline reward information—which our approach does not require. Furthermore, we demonstrate that the GP behavioral policy's predictive variance is crucial for KL-regularized objectives to learn good online policies from expert demonstrations. Finally, we perform ablation studies that illustrate that non-parametric GP behavioral reference policies also outperform parametric behavioral reference policies with improved uncertainty quantification, such as deep ensembles and Bayesian neural

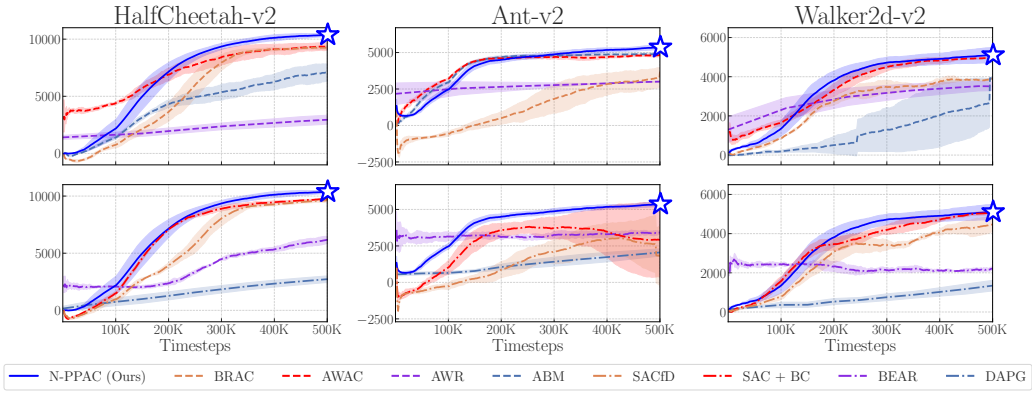


Figure 4: Comparison of N-PPAC (ours) vs. previous baselines on standard MuJoCo benchmark tasks. **Top**: KL-based methods (dashed lines), **Bottom**: Non-KL-based methods (dash-dotted lines). Both top and bottom plots include N-PPAC (**blue**). BRAC uses the same actor–critic algorithm as N-PPAC, but uses a parametric behavioral policy, and results in slower learning and worse final performance.

networks (BNNs) with Monte Carlo dropout, and that the difference between non-parametric and parametric models is exacerbated the fewer expert demonstrations are available. We use the expert data from Nair et al. [27], every experiment uses six random seeds, and we use a fixed KL-temperature for each environment class. For further implementation details, see Appendix C.2.

5.1 Environments

MuJoCo locomotion tasks. We evaluate N-PPAC on three representative tasks: "Ant-v2", "HalfCheetah-v2", and "Walker2d-v2". For each task, we use 15 demonstration trajectories collected by a pre-trained expert, each containing 1,000 steps. The behavioral policy is specified as the posterior distribution of a GP with a squared exponential kernel, which is well-suited for modeling smooth functions.

Dexterous hand manipulation tasks. Real-world robot learning is a setting where human demonstration data is readily available, and many deep RL approaches fail to learn efficiently. We study this setting in a suite of challenging dexterous manipulation tasks [35] using a 28-DoF five-fingered simulated ADROIT hand. The tasks simulate challenges common to real-world settings with high-dimensional action spaces, complex physics, and a large number of intermittent contact forces. We consider two tasks in particular: in-hand rotation of a pen to match a target and opening a door by unlatching and pulling a handle. We use binary rewards for task completion, which is significantly more challenging than the original setting considered in Rajeswaran et al. [35]. 25 expert demonstrations were provided for each task, each consisting of 200 environment steps which are not fully optimal but do successfully solve the task. The behavioral policy is specified as the posterior distribution of a GP with a Matérn kernel, which is more suitable for modeling non-smooth data.

5.2 Results

On MuJoCo environments, KL-regularized RL with a non-parametric behavioral policy consistently outperforms all related methods across all three tasks, successfully accelerating learning from offline data, as shown in Figure 4. Most notably, it outperforms methods such as AWAC [27]—the previous state-of-the-art—which attempts to eschew the problem of learning behavioral policies but instead uses an implicit constraint. Our approach, N-PPAC, exhibits an increase in stability and higher returns compared to comparable methods such as ABM and BRAC that explicitly regularize the online policy against a parametric behavioral policy and plateau at suboptimal performance levels as they are being forced to copy poor actions from the behavioral policy away from the expert data. In contrast, using a non-parametric behavioral policy allows us to avoid such undesirable behavior.

On dexterous hand manipulation environments, KL-regularized RL with a non-parametric behavioral policy performs on par or outperforms all related methods on both tasks, as shown in Figure 5. Most notably, on the door opening task, it achieves a stable success rate of 90% within only 100,000 environment interactions For comparison, AWAC requires 4× as many environment interactions to

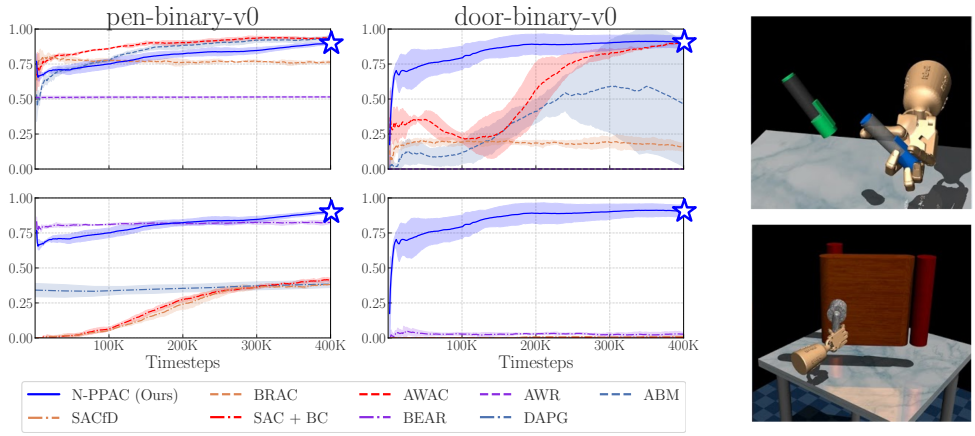


Figure 5: Left & Center: Comparison of N-PPAC (ours) vs. previous baselines on dexterous hand manipulation tasks. **Top**: KL-based methods (dashes), **Bottom**: Non-KL-based methods (dots and dashes). Both top and bottom plots include N-PPAC (**blue**). **Right**: The pen-binary-v0 (top) and door-binary-v0 (bottom) environments.

achieve the same performance and is significantly less stable, while most other methods fail to learn any meaningful behaviors.

Alternative divergence metrics underperform KL-regularization. KL-regularized RL with a non-parametric behavioral policy consistently outperforms methods that use alternative divergence metrics, as shown in the bottom plots of Figures 4 and 5.

5.3 Can the Pathology Be Fixed by Improved Parametric Uncertainty Quantification?

To assess whether the success of non-parametric behavioral reference policies is due to their predictive variance estimates—as suggested by Proposition 1—or due to better generalization from their predictive means, we perform an ablation study on the predictive variance of the behavioral policy. To isolate the effect of the predictive variance on optimization, we perform online training using behavioral policies with different predictive variance functions (parametric and non-parametric) and identical mean functions, which we set to be the predictive mean of the GP posterior (which achieves a success rate of ~80%). If the pathology identified in Proposition 1 can be remedied by commonly used parametric uncertainty quantification methods, we would expect the parametric and non-parametric behavioral policy variance functions to result in similar online policy success rates. We consider the challenging "door-binary-v0" environment for this ablation study.

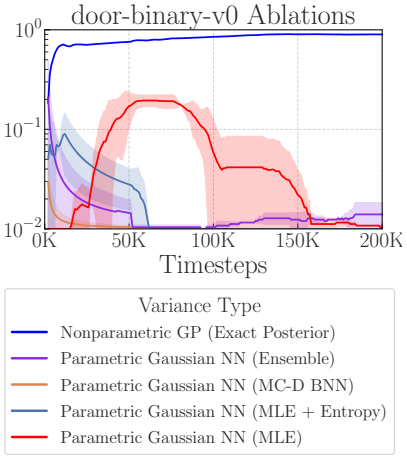


Figure 6: Post-online training success rates with different behavioral policy variance functions.

Parametric uncertainty quantification is insufficient. Figure 5 shows that parametric variance functions result in online policies that only achieve success rates of up to 20% and eventually deteriorate, whereas the non-parametric variance yields an online policy that achieves a success rate of nearly 100%. This finding shows that commonly used uncertainty quantification methods, such as deep ensembles or BNNs with Monte Carlo dropout, do not generate sufficiently well-calibrated uncertainty estimates to remedy the pathology, and better methods may be needed [9, 39, 41].

Lower-bounding the predictive variance does not remedy the pathology. The predictive variance of all MLE-based and ensemble behavioral reference policies in all experiments are bounded away from zero at a minimum value of $\approx 10^{-2}$. Hence, setting a floor on the variance is not sufficient to prevent pathological training dynamics. This result further demonstrates the importance of accurate predictive variance estimation in allowing the online policy to match expert actions in regions of the state space with low behavioral policy predictive variance and explore elsewhere.

5.4 Can a Single Expert Demonstration Be Sufficient to Accelerate Online Training?

To assess the usefulness of non-parametric behavioral reference policies in settings where only few expert demonstrations are available, we investigate whether the difference in performance between online policies trained with non-parametric and parametric behavioral reference policies, respectively, is exacerbated the fewer expert demonstrations are available. To answer this question, we consider the "HalfCheetah-v2" environment and compare online policies trained with different behavioral reference policies—non-parametric GPs, deep ensembles, and BNNs with Monte Carlo dropout—estimated either from 15 expert demonstrations (i.e., 15 state–action trajectories, containing 15,000 samples) or from a single expert demonstration (i.e., a single state–action trajectory, containing 1,000 samples).

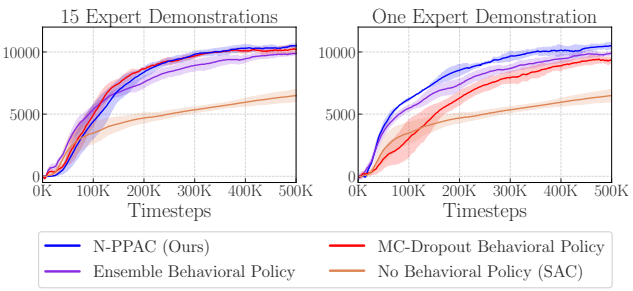


Figure 7: Returns during online training with different behavioral policies and varying amounts of expert demonstration data on "HalfCheetah-v2".

A single expert demonstration is sufficient for non-parametric behavioral reference policies. Figure 7 shows the returns for online policies trained with behavioral reference policies estimated from the full dataset (top plot) and from only a single expert state–action trajectory (bottom plot). On the full dataset, we find that all three methods are competitive and improve on the prior state-of-the-art but that the GP behavioral policy leads to the highest return. Remarkably, non-parametric GP behavioral policies perform just as well with only a single expert demonstration as with all 15 (i.e., with 1,000 data points, instead of 15,000 data points). These results further emphasizes the usefulness of non-parametric behavioral policies when accelerating online training with expert demonstrations—even when only very few expert demonstrations are available.

5.5 Are Non-Parametric GP Behavioral Reference Policies Too Computationally Expensive?

Table 1 presents the time complexity of KL-regularized RL under non-parametric GP and parametric neural network behavioral reference policies, as measured by the average time elapsed per epoch on the "door-binary-v0" and "HalfCheetah-v2" environments. One epoch of online training on "door-binary-v0" and "HalfCheetah-v2" requires computing the KL divergence over 1,000 mini-batches of size 256 and 1,024, respectively. The time complexity of evaluating the log-density of a GP behavioral reference policy—needed for computing gradients of the KL divergence during online training—scales quadratically in the number of training data points and linearly in the dimensionality of the state and action space, respectively. As can be seen in Table 1, non-parametric GP behavioral reference policies only lead to a modest increase in the time needed to complete one epoch of training while resulting in significantly improved performance as shown in Figures 4 and 5.

Table 1: Time per epoch under different behavioral reference policies for expert demonstration data of varying size computed on a GeForce RTX 3080 GPU. The first and second value in each entry of the table give the time required when using a single parametric neural network and a GP behavioral reference policy, respectively.

Dataset	1,000 Data Points	5,000 Data Points	15,000 Data Points
HalfCheetah-v2	12.00s / 16.06s	11.59s / 18.31s	12.00s / 46.54s
door-binary-v0	19.62s / 23.78s	19.62s / 33.62s	-

6 Conclusion

We identified a previously unrecognized pathology in KL-regularized RL from expert demonstrations and showed that this pathology can significantly impede and even entirely prevent online learning. To remedy the pathology, we proposed the use of non-parametric behavioral reference policies, which we showed can significantly accelerate and improve online learning and yield online policies that (often significantly) outperform current state-of-the-art methods on challenging continuous control tasks. We hope that this work will encourage further research into better model classes for deep reinforcement learning algorithms, including and especially for reinforcement from image inputs.

Acknowledgments and Disclosure of Funding

We thank Ashvin Nair for sharing his code and results, as well as for providing helpful insights about the dexterous hand manipulation suite. We also thank Clare Lyle, Charline Le Lan, and Angelos Filos for detailed feedback on an early draft of this paper, Avi Singh for early discussions about behavioral cloning in entropy-regularized RL, and Tim Pearce for a useful discussion on the role of good models in RL. TGJR and CL are funded by the Engineering and Physical Sciences Research Council (EPSRC). TGJR is also funded by the Rhodes Trust and by a Qualcomm Innovation Fellowship. We gratefully acknowledge donations of computing resources by the Alan Turing Institute.

References

[1] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[2] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 619–629. PMLR, 18–24 Jul 2021.

[3] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189. JMLR Workshop and Conference Proceedings, 2011.

[4] Ivan Bratko, Tanja Urbancic, and Claude Sammut. Behavioural cloning: phenomena, results and problems. *IFAC Proceedings Volumes*, 28(21):143–149, 1995.

[5] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstrations through shaping. In *Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015.

[6] Catherine Cang, Aravind Rajeswaran, Pieter Abbeel, and Michael Laskin. Behavioral priors and dynamics models: Improving performance and domain transfer in offline RL, 2021.

[7] Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 179–186, Madison, WI, USA, 2012. Omnipress.

[8] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[9] Sebastian Farquhar, Michael A. Osborne, and Yarin Gal. Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1352–1362. PMLR, 26–28 Aug 2020.

[10] Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[11] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[12] CW Groetsch. The theory of Tikhonov regularization for Fredholm equations. *Boston Pitman Publication*, 1984.

[13] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

[14] Hado V. Hasselt. Double Q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621, 2010.

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[16] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020.

[17] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[18] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012.

[19] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems 32*, pages 11784–11794, 2019.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413, 2017.

[21] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

[22] 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 *ICLR (Poster)*, 2016.

[23] Cong Lu, Philip J. Ball, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts. Revisiting design choices in model-based offline reinforcement learning, 2021.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[25] Kevin P. Murphy. *Machine learning: A Probabilistic Perspective*. MIT Press, 2013.

[26] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299, 2018.

[27] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets, 2020.

[28] Nicolás Navarro-Guerrero, Cornelius Weber, Pascal Schroeter, and Stefan Wermter. Real-world reinforcement learning for autonomous humanoid robot docking. *Robotics and Autonomous Systems*, 60(11):1400–1407, 2012.

[29] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[30] Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Krzysztof Choromanski, Anna Choromanska, and Michael Jordan. Learning to score behaviors for guided policy optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7445–7454. PMLR, 13–18 Jul 2020.

[31] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

[32] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning*, pages 280–291. Springer, 2005.

[33] Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, December 2005. ISSN 1532-4435.

[34] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[35] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018.

[36] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[37] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

[38] Michael T Rosenstein, Andrew G Barto, Jennie Si, Andy Barto, and Warren Powell. Supervised actor-critic reinforcement learning. *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*, pages 359–380, 2004.

[39] Tim G. J. Rudner, Zonghao Chen, and Yarin Gal. Rethinking function-space variational inference in Bayesian neural networks. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

[40] Tim G. J. Rudner, Vitchyr H. Pong, Rowan Thomas McAllister, Yarin Gal, and Sergey Levine. Outcome-driven reinforcement learning via variational inference. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=4bzanicqvy8.

[41] Tim G. J. Rudner, Freddie Bickford Smith, Qixuan Feng, Yee Whye Teh, and Yarin Gal. Continual learning via function-space variational inference. In *ICML Workshop on Theory and Foundations of Continual Learning*, 2021.

[42] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[43] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

[44] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[45] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto, editors, *Algorithmic Learning Theory*, pages 13–31, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[46] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[47] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[48] Emanuel Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1369–1376. MIT Press, 2007.

[49] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. Improving deterministic uncertainty estimation in deep learning for classification and regression, 2021.

[50] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact Gaussian processes on a million data points. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[51] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[52] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020.

[53] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization, 2021.

Limitations and Future Work

Our algorithm, N-PPAC, relies on the well-calibrated behavioral priors we derive from Gaussian processes. However, this is also the key limitation—we are restricted to expert datasets that are amenable to training and performing exact inference on with a GP. Although studies have demonstrated the feasibility of exact inference on datasets up to a million datapoints (Wang et al., 2019), this requires multi-GPU parallelization and careful partitioning. We highlight two scalable directions to applying GPs to larger and higher-dimensional datasets. The first direction is performing approximate inference with inducing points (Leibfried et al., 2022). These methods replace the original training data with pseudo-training examples (inducing points) which allows one to trade off accuracy against run-time. The second direction is deep GPs (Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017), multi-layer generalizations of GPs, that have been proven to work for image classification and datasets on the order of billions of datapoints.

More generally, our paper draws attention to the use of different model classes across reinforcement learning where the predominant choice is a neural network. We showed how accurate uncertainty quantification can significantly improve regularization from expert demonstrations. A strongly related area of reinforcement learning which also relies on uncertainty quantification relative to a static dataset is offline reinforcement learning (Brandfonbrener et al., 2022; Kidambi et al., 2020; Yu et al., 2020b). These methods typically use deep ensembles to mitigate distribution shift and would be a perfect next step for investigating alternative model classes.

Furthermore, an interesting recent alternative to GPs is function-space variance inference (Rudner et al., 2022) which proposes a scalable Bayesian neural network approach to inferring a posterior distribution over functions. Finally, we note that our method also relied on the predictive mean of the GP behavioral prior being informative. This points to the efficacy of kernel-based methods (Bierens, 1994) for behavioral cloning, even in scenarios where accurate predictive variance estimates are not required. Finally, our paper makes the assumption that the expert

demonstrations are unimodal conditional on the state, as our supervised objectives will fit the mean of the expert policy. One way to overcome this restriction would be to consider likelihood models that are able to fit multiple modes, for example, conditional diffusion models as demonstrated by Pearce et al. (2023).

6

Revisiting Design Choices in Offline Model-Based Reinforcement Learning.

In this chapter, we move to the fully offline setting with no online interaction. We begin by focusing on model-based methods, which help mitigate distribution shift by generating synthetic on-policy data. As discussed in Section 2.3.2, contemporary algorithms typically construct pessimistic MDPs which are based on the following theoretical lower bound on the true return

$$J_M(\pi) \geq \mathbb{E}_{\pi,\widehat{P}}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t)-\gamma|G^{\pi}_{\widehat{M}}(s_t,a_t)|\right)\right] \tag{6.1}$$

where $G^{\pi}_{\widehat{M}}(s,a)$ is a measure of discrepancy between the true and model dynamics as measured by the value function. However, in practice, we do not have access to the true dynamics, making this penalty difficult to compute. Therefore, existing methods exhibit a breakdown between theory and practice, and instead, the penalty is implemented based on estimated model uncertainty. This has spawned a variety of heuristics (Kidambi et al., 2020; Yu et al., 2020b), with little to no comparison between differing approaches.

Since the datasets in offline RL can be up to two million datapoints, the techniques described in the previous chapter are no longer applicable, and instead, uncertainty

is usually approximated using deep ensembles. To motivate our investigation, we first consider the form of two commonly used uncertainty metrics[1]:

- **Max Aleatoric (Yu et al., 2020b):** $\max_{i=1,\ldots,N} ||\Sigma_\phi^i(s,a)||_{\mathrm{F}}$, which computes a maximum over the variance heads of the ensemble.
- **Max Pairwise Difference (Kidambi et al., 2020):** $\max_{1\leq i,j\leq N} ||\mu_\phi^i(s,a) - \mu_\phi^j(s,a)||_2$, maximizing over pairwise differences in the mean predictions.

This immediately raises two questions: first, how do the max operator and number of ensemble members N skew the above penalties? Second, why haven't standard uncertainty metrics from supervised learning such as the original mixture variance below been used?

- **Ensemble Variance (Lakshminarayanan et al., 2017):** $\Sigma^*(s,a) = \frac{1}{N}\sum_{i=1}^N(\Sigma_\phi^i(s,a) + \mu_\phi^i(s,a)^2) - \mu^*(s,a)^2$ is the variance of the mixture distribution where $\mu^*(s,a) = \frac{1}{N}\sum_{i=1}^N \mu_\phi^i(s,a)$ is the mean of the means.

These questions are particularly pertinent as the deep ensembles used in offline RL are very small ($N = 7$) compared to typical sizes in supervised learning. Furthermore, uncertainty quantification with the ensemble variance has been shown to improve with an increasing number of ensemble members. Even more fundamentally, how well do these penalties capture errors in model predictions, particularly as we increase the rollout horizon which is low ($k = 5$) in existing methods?

In this paper, we compare these heuristics, design novel protocols to investigate the interactions of all these hyperparameters, and elucidate a design space for offline model-based reinforcement learning. We then show if we are allowed *online evaluation at the end of each training run*, that selecting these key hyperparameters using Bayesian Optimization (introduced in Section 2.1.5) produces superior configurations that are vastly different from those currently used in existing hand-tuned state-of-the-art methods, and result in drastically stronger performance. For each environment, we allow a budget of 100 offline training runs together with an

[1]We note in comparison to the Chapter 5, μ and Σ refer to the dynamics model rather than the behavioral prior.

online evaluation at the end. We find that these findings may then be distilled back into a more realistic offline scenario by choosing the best two hyperparameter configurations (which is less than usual in offline RL), leading to a 69% improvement over the baseline. Particular highlights of our analysis include: more canonical forms of uncertainty estimation are better calibrated with model error and can substitute for hand-designed heuristics, and much higher rollout horizons may be used given high enough penalty weight.

Cong Lu*, Philip J. Ball*, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts. Revisiting Design Choices in Offline Model Based Reinforcement Learning. **In ICLR (Spotlight), 2022.**

REVISITING DESIGN CHOICES IN OFFLINE MODEL-BASED REINFORCEMENT LEARNING

Cong Lu*, **Philip J. Ball***, **Jack Parker-Holder, Michael A. Osborne, Stephen J. Roberts**
Department of Engineering
University of Oxford

ABSTRACT

Offline reinforcement learning enables agents to leverage large pre-collected datasets of environment transitions to learn control policies, circumventing the need for potentially expensive or unsafe online data collection. Significant progress has been made recently in offline model-based reinforcement learning, approaches which leverage a learned dynamics model. This typically involves constructing a probabilistic model, and using the model uncertainty to penalize rewards where there is insufficient data, solving for a *pessimistic* MDP that lower bounds the true MDP. Existing methods, however, exhibit a breakdown between theory and practice, whereby pessimistic return ought to be bounded by the *total variation distance* of the model from the true dynamics, but is instead implemented through a penalty based on estimated *model uncertainty*. This has spawned a variety of uncertainty heuristics, with little to no comparison between differing approaches. In this paper, we compare these heuristics, and design novel protocols to investigate their interaction with other hyperparameters, such as the number of models, or imaginary rollout horizon. Using these insights, we show that selecting these key hyperparameters using Bayesian Optimization produces superior configurations that are vastly different to those currently used in existing hand-tuned state-of-the-art methods, and result in drastically stronger performance.

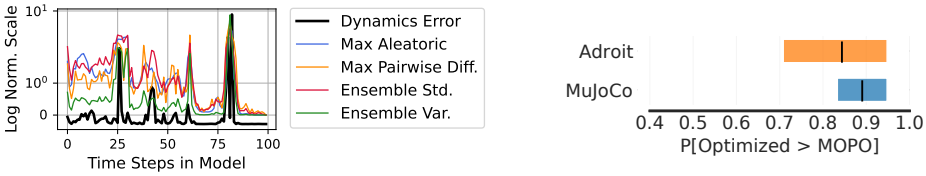
1 INTRODUCTION

In offline (or batch) reinforcement learning (RL) (Ernst et al., 2005; Levine et al., 2020), the goal is to leverage offline datasets of transitions in an environment to train a policy that transfers to an online task. This could have vast implications for using RL in real-world settings, as agents can make use of ever-increasing amounts of data without the need for an accurate simulator, while also avoiding expensive and potentially even unsafe exploration in the environment.

Model-based reinforcement learning (MBRL) has recently shown promise in this paradigm, obtaining state-of-the-art performance on offline RL benchmarks (Kidambi et al., 2020; Yu et al., 2021), improving upon powerful model-free approaches (e.g. Kumar et al. (2020)). MBRL works by training a dynamics model from the offline data, then optimizing a policy using imaginary rollouts from the model. This allows the agent to learn from on-policy experience, as the model is agnostic to the policy used to generate data, making it possible to achieve high returns using data collected from even a random policy. Furthermore, recent work has demonstrated the utility of world models *beyond* maximizing return, such as generalizing to unseen variations in an environment (Ball et al., 2021), transferring to new tasks (Yu et al., 2020), and learning with safety constraints (Argenson & Dulac-Arnold, 2021). Therefore, the case for MBRL in offline RL is clear: not only does it represent state-of-the-art in terms of performance, but it also provides the opportunity to maximize the signal in the offline data to generalize onto tasks beyond those encoded by the behavior policy. This is crucial for offline RL to be useful for real-world tasks (Dulac-Arnold et al., 2021), where there will inevitably be differences between the data and desired task.

However, a common failure mode of MBRL is when policies exploit the model in parts of the state-action space where the model is inaccurate. Thus, naïve application of MBRL to offline data

*Joint first authors. Correspondence: cong.lu@stats.ox.ac.uk, ball@robots.ox.ac.uk



(a) **Uncertainty vs. True Errors** (b) **Probability of Improvement**

Figure 1: **a)** The variation of different uncertainty penalties against *true* dynamics error during a model rollout of Hopper Medium-Expert. The canonical ensemble variance penalty most closely fits the true dynamics error. **b)** Tuning key hyperparameters (an approach we call Optimized) can lead to large gains over state-of-the-art methods (MOPO) on the D4RL benchmark, as we show in this summary using `rliable` (Agarwal et al., 2021).

can result in suboptimal performance. To prevent this, concurrent works (Yu et al., 2020; Kidambi et al., 2020) have approached the problem by training a policy in a *pessimistic* MDP (P-MDP). The P-MDP lower bounds the true MDP, and discourages the policy from regions where there is large discrepancy between the true and learned dynamics; this often provides a theoretical guarantee of improvement over cloning the behavior policy that generated the offline data. This is made practically possible by adding a penalty correlated with the uncertainty in the dynamics model. However, while these recent successes are similar in principle, in practice they differ in a series of design choices. First and foremost, they make use of different heuristics to measure model uncertainty, in some cases deviating from simpler metrics which are more consistent with the theory.

In this paper, we conduct a rigorous investigation into a series of these design choices. We begin by focusing on the choice of uncertainty metric, comparing both recent state-of-the-art offline approaches (Kidambi et al., 2020; Yu et al., 2020; Rafailov et al., 2020) with additional metrics used in the online setting (Ball et al., 2020; Pan et al., 2020; Cowen-Rivers et al., 2022). We also explore the interaction with a series of other hyperparameters, such as the number of models and imaginary rollout length. Interestingly, the relationship between these variables and model uncertainty varies significantly depending on the choice of uncertainty penalty. Furthermore, we compare these uncertainty heuristics under new evaluation protocols that, for the first time, capture the specific covariate shift induced by model-based RL. This allows us to assess calibration to model exploitation in MBRL, observing that some existing penalties are surprisingly successful at capturing the errors in predicted dynamics, as seen in Fig. 1a (see App. D for details). Then, using the insights gained from sections 4 and 5, we then achieve a **43%** gain over a previously grid-searched method by using a *single hyperparameter* value across all environments. We then jointly fine-tune our identified key variables using a powerful Bayesian Optimization algorithm (Wan et al., 2021) and find the simpler uncertainty measures can provide state-of-the-art results in continuous control offline benchmarks, and that the chosen optimal hyperparameters continue to align with our analysis. Finally, we rigorously confirm the aggregate improvement of our results using the `rliable` framework (Agarwal et al., 2021) in Fig. 1b, and show that the improvements over existing methods are significant (see App. H for details). This work is intended to benefit both researchers and practitioners in offline RL. Our main findings include:

- **Longer horizon rollouts with larger penalties can improve existing methods.** Contrary to common wisdom, conducting significantly longer rollouts inside the model, coupled with larger uncertainty penalties, typically improves performance.
- **Penalties that use canonical forms of uncertainty estimation achieve better correlation with OOD measures.** The uncertainty estimation approach of Lakshminarayanan et al. (2017) often outperforms the penalty from state-of-the-art methods (Yu et al., 2020; Kidambi et al., 2020). We observe that the ensemble standard deviation is statistically strikingly similar to that used in Kidambi et al. (2020), but has improved correlation and scaling behavior.
- **Uncertainty is more correlated with *dynamics error* than *distribution shift*.** We find that successful penalties measure the discrepancy in dynamics, and can in fact assign high certainty to data far away from the offline data.

2 Related Work

Two recent works concurrently demonstrated the effectiveness of model-based reinforcement learning (MBRL) in the offline setting. MOPO (Yu et al., 2020) follows the successful online RL algorithm MBPO (Janner et al., 2019) but trains inside a *conservative MDP*, penalizing the reward based on the maximum aleatoric uncertainty over the ensemble members. MOReL (Kidambi et al., 2020) achieves

even stronger performance, penalizing the rewards by a penalty based on the maximum pair-wise difference in ensemble member predictions. For pixel-based tasks, LOMPO (Rafailov et al., 2020) also proposed a novel penalty, using the variance of ensemble log-likelihoods. Outside the offline setting, probabilistic dynamics models leveraging uncertainty have underpinned a series of successes (Chua et al., 2018; Kurutach et al., 2018; Buckman et al., 2018; Pan et al., 2020; Pacchiano et al., 2021). Uncertainty can also be measured in MBRL without the use of neural networks (Deisenroth & Rasmussen, 2011), although these methods tend to be harder to scale and thus lack widespread use.

Effective hyperparameter selection in RL has been shown to be crucial to the success of popular algorithms (Engstrom et al., 2020; Andrychowicz et al., 2021). This becomes even more challenging in MBRL with additional hyperparameters/design-choices for the dynamics model. Recent work has shown that carefully optimizing these hyperparameters for online MBRL can significantly improve performance, with the tuned agent breaking the MuJoCo simulator (Zhang et al., 2021). In contrast, we focus on the offline setting, and investigate parameters specifically related to uncertainty estimation. Previous work studied the impact of hyperparameters in offline RL (Paine et al., 2020), finding offline RL algorithms to be brittle to hyperparameter choices. However, unlike our work they only consider model-free approaches, whereas we specifically investigate *model-based* offline algorithms. Abbas et al. (2020) investigates the impact of different uncertainty estimation methods in online MBRL; they too find penalizing with combined aleatoric and epistemic uncertainty improves performance.

Our work also relates to the rich literature on *deep ensembles* (Lakshminarayanan et al., 2017), which train multiple deep neural networks with different initializations and dataset orderings, and generally outperform variational Bayes methods (Mackay, 1992; Blundell et al., 2015). Achieving effective calibration with neural networks is notoriously difficult (Guo et al., 2017; Kuleshov et al., 2018; Maddox et al., 2019), and furthermore we require calibration under co-variate shift (Ovadia et al., 2019), as the policy learned in the model will likely deviate from the behavior policy that generated the offline data. Recent work has highlighted this issue in offline RL (Kumar et al., 2020; Yu et al., 2021) and has reported superior performance when eschewing model uncertainty entirely, and instead performing "conservative" Q-updates. However, it is unclear if this improvement is due to poor uncertainty calibration, implementation details, or a limitation in the pessimistic-MDP formulation.

3 Background

All of the methods we investigate in this paper model the environment as a Markov Decision Process (MDP), defined as a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(s'|s,a)$ the transition dynamics, $R(s,a)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in (0,1)$ the discount factor. The goal is to optimize a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

In *offline RL*, the policy is not deployed in the environment until test time. Instead, the algorithm only has access to a static dataset $\mathcal{D}_{env} = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^{J}$, collected by one or more behavioral policies π_b. Following the notation in Yu et al. (2020) we refer to the distribution from which $\mathcal{D}_{env}$ was sampled as the *behavioral distribution*. The canonical approach in offline MBRL is to train an ensemble of N probabilistic dynamics models (Nix & Weigend, 1994). These usually learn to predict both the next state s_{t+1} and reward r_t from a state-action pair, and are trained on $\mathcal{D}_{env}$ using supervised learning. Concretely, each of the N models output a Gaussian $\widehat{P}^i_\phi(s_{t+1}, r_t|s_t, a_t) = \mathcal{N}(\mu^i_\phi(s_t,a_t), \Sigma^i_\phi(s_t,a_t))$ parameterized by ϕ. The resulting learned dynamics model $\widehat{P}$ and reward model $\widehat{R}$ define a *model MDP* $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \widehat{R}, \rho_0, \gamma)$. To train the policy, we use k-step rollouts inside $\widehat{M}$ to generate trajectories (Sutton, 1991).

To prevent policy exploitation in a model, a pessimistic MDP (P-MDP) is constructed by lower bounding the true-expected return, $\eta_M(\pi)$, using some error between the true and estimated models. For instance, in Yu et al. (2020), the authors show that a lower bound on the return can be established by penalizing the reward by a measure that corresponds to estimated model error:

$$\eta_M(\pi) \geq \mathop{\mathbb{E}}_{(s,a)\sim\rho^\pi_{\hat{P}}} \left[R(s,a) - \gamma|G^\pi_{\hat{M}}(s,a)|\right] \tag{1}$$

where $\rho^\pi_{\hat{P}}$ represents transitioning under the dynamics model $\hat{P}$ and policy π. Several potential choices for $|G^\pi_{\hat{M}}(s,a)|$ are proposed, including an upper bound based on the total variation distance

between the learned and true dynamics. However, for their practical algorithm, the authors elect to use a heuristic based on impressive empirical results. Concurrent to MOPO, MOReL (Kidambi et al., 2020) in theory constructs a P-MDP by augmenting a standard MDP with a negative valued absorbing state that is transitioned to when total variation distance between true and learned dynamics is exceeded. They show that a policy learned in this P-MDP exceeds simple behavior cloning. However, while dynamics-based total variation distance has desirable theoretical properties, the practical algorithm relies on another heuristic to approximate this quantity. This motivates the study of penalties used, as well as other under-used candidates, and their overall effectiveness.

4 Uncertainty Penalty

The key idea underpinning recent success in offline MBRL is the introduction of a P-MDP, penalized by some uncertainty penalty. The theory dictates this should be some distance measure between the true and predicted dynamics. Of course, this cannot be truly estimated without access to an oracle, so a proxy for this quantity is constructed instead based on uncertainty heuristics. In this paper, we compare the following uncertainty heuristics, from recent works in both offline and online MBRL:

Max Aleatoric (Yu et al., 2020): $\max_{i=1,\dots,N} ||\Sigma^i_\phi(s,a)||_{\mathrm{F}}$, which corresponds to the maximum aleatoric error, computed over the variance heads of the model ensemble.
Max Pairwise Diff (Kidambi et al., 2020): $\max_{i,j} ||\mu^i_\phi(s,a) - \mu^j_\phi(s,a)||_2$, which corresponds to the pairwise maximum difference of the ensemble predictions.
LL Var (Log-Likelihood Variance) (Rafailov et al., 2020)): $\mathrm{Var}(\{\log \widehat{P}^i_\phi(s'|s,a), i = 1,\dots,N\})$, where s' is a next state sampled from a single ensemble member. We evaluate its log-likelihood under each ensemble member and take the variance.
LOO KL (Leave-One-Out KL Divergence (Pan et al., 2020): $D_{\mathrm{KL}}[\widehat{P}_{\phi_i}(\cdot|s,a)||\widehat{P}_{\phi_{-i}}(\cdot|s,a)]$, which corresponds to the KL divergence between the Gaussian parameterized by a randomly selected ensemble member, and the aggregated Gaussian of the remaining ensemble members.
Ensemble Standard Deviation/Variance (Lakshminarayanan et al., 2017): The variance is given as: $\Sigma^*(s,a) = \frac{1}{N}\sum_i^N ((\Sigma^i_\phi(s,a))^2 + (\mu^i_\phi(s,a))^2) - (\mu^*(s,a))^2$ where μ^* is the mean of the means ($\mu^*(s,a) = \frac{1}{N}\sum_i^N \mu^i_\phi(s,a)$). This corresponds to a combination of epistemic and aleatoric model uncertainty. This is surprisingly under-utilized in offline MBRL, and is a canonical method of uncertainty estimation used in the Bayesian inference literature (Ovadia et al., 2019; Filos et al., 2019; Scalia et al., 2020). We choose to evaluate both standard deviation (the square root of the above) and variance, as this will provide intuition about the importance of penalty distribution *shape*.

These can all be computed using the output from an ensemble of probabilistic dynamics models (Lakshminarayanan et al., 2017), so we are able to compare them in a controlled manner.

4.1 How Well Do Ensemble Penalties Detect Out of Distribution Errors?

We begin by assessing how well uncertainty penalties correlate with next state MSE (we justify the MSE under deterministic dynamics in App. A.2). This is crucial in penalizing the policy from visiting parts of the state-action space where the model is inaccurate, and therefore exploitable. Using D4RL (Fu et al., 2021a), we train models on each dataset, then evaluate them on *other datasets* from the same environment, but collected under *different* policies. These form our "Transfer" experiments as they directly measure the ability of uncertainty penalties at detecting errors on *unseen* data. We compare the penalties against true MSE for a variety of settings in App. A.3, and summarize this in the "Transfer" column of Table 1. We measure Spearman rank (ρ) and Pearson bivariate (r) correlations, and justify their use in App. A.1. Full details of all experiments and hyperparameters are given in App. G. We will analyze these results in detail in the next section, after introducing a novel protocol for assessing our penalties under the out-of-distribution (OOD) data induced by model exploitation.

4.2 How Do These Perform During an Imaginary Rollout?

We additionally design an experiment aimed at capturing the OOD data *generated by the actual offline MBRL process*, which we call our "True Model-Based" experiments. First, we train a set of policies with 4 different starting seeds *without* a penalty inside the model for 500 iterations. We

Table 1: Correlation statistics of penalties against true mean-sq. model error, averaged over all datasets (i.e., Random through to Expert) showing ± 1 SD over 12 seeds. The best in each column is **bolded**. The ensemble penalties generally perform best.

	Transfer				True Model-Based			
	HalfCheetah		Hopper		HalfCheetah		Hopper	
Penalty	ρ	r	ρ	r	ρ	r	ρ	r
Max Aleatoric	0.78±0.00	0.55±0.01	0.71±0.01	0.41±0.01	0.58±0.01	0.42±0.01	0.73±0.03	0.48±0.01
Max Pairwise Diff.	0.79±0.01	0.62±0.00	0.77±0.00	0.57±0.00	0.58±0.01	**0.52**±0.01	0.75±0.02	**0.55**±0.02
Ens. Std.	**0.82**±0.01	0.64±0.01	**0.79**±0.00	0.56±0.00	**0.61**±0.01	**0.52**±0.00	**0.79**±0.02	**0.55**±0.02
Ens. Var.	**0.82**±0.01	**0.67**±0.00	**0.79**±0.00	**0.59**±0.00	0.60±0.01	0.49±0.01	0.77±0.02	**0.55**±0.02
LL Var.	0.13±0.05	0.14±0.02	0.36±0.04	0.12±0.02	0.04±0.16	0.07±0.06	0.50±0.02	0.16±0.02
LOO KL	0.03±0.02	0.11±0.02	0.11±0.02	0.08±0.02	-0.02±0.12	0.06±0.06	0.22±0.02	0.10±0.02

then measure the difference between the return predicted by the model over a rollout, and the true return in the real environment. We define a policy to be "exploitative" if the model significantly *over-estimates* the return compared to the true return. It is these exploitative policies that induce the types of extrapolation errors which cause MBRL methods to fail in the offline setting. It is therefore important that the penalty is able to accurately determine when the model is being exploited in this way. We use a subset of the 5 most exploitative policies to generate trajectories in the model, and record the uncertainty predicted by each penalty at every time step. To generate the True Model-Based data, we then "replay" these trajectories in the true environment, loading the state and action taken in the model into the environment, and record the "true" next state according to the MuJoCo simulator (Todorov et al., 2012). True Model-Based therefore calculates the MSE between the *predicted* and *actual* next states. Table 1 summarizes the results from both the Transfer and True Model-Based experiments. Additional details are provided in App. D along with full correlation plots in App. A.3.

We are now in a position to analyze the results in Table 1. It is immediately obvious that the LOO KL and LL Var penalties have very weak correlation with MSE. We believe this is because LL Var relies on likelihood statistics, which are notoriously sensitive; it was designed for use with a KL-regularized latent state space model which has well-behaved dynamics. Regarding LOO KL, we note that this penalty was designed for the online setting with significantly less data, and becomes quite uncorrelated in this larger data setting. This advocates penalties that are less reliant on distributional information concerning the separate Gaussians in the ensemble, as such penalties appear sensitive to the quality of their estimated distributions. We observe that Max Aleatoric, Max Pairwise Diff and the Ensemble penalties perform broadly similarly despite their different analytical forms; Ensemble measures do however exhibit noticeably higher rank correlation. We also observe a significant performance loss between the Transfer and True Model-Based HalfCheetah settings, with the latter being relatively poor. This implies further work is needed to develop penalties that can successfully detect the type of dynamics discrepancies that actually arise in offline MBRL. Finally, we observe that despite the similar rank correlations ρ, the bivariate correlations r can vary considerably, and observe from the scatter plots that Max Aleatoric exhibits low kurtosis, having large penalty values "bunched" at its extreme; we provide 3rd and 4th order moment statistics to facilitate shape comparisons in App. C.

5 Key Hyperparameters in Offline MBRL

5.1 How Many Models Do We Need?

At present the number of models used has not been discussed since MBPO, which trains seven probabilistic dynamics models of the same architecture (with different initializations), using only the top five models based on validation accuracy (referred to as "Elites" in the Evolutionary community, e.g. Mouret & Clune (2015)). The reason or justification for this is not discussed, but it has seemingly been adopted in the wider MBRL setting (Shen et al., 2020; Omer et al., 2021; Pineda et al., 2021). However, offline RL is a totally different paradigm, where it is possible that access to compute is less of a bottleneck and it may be preferable to use more models to extract the most signal possible from the static dataset. Inevitably, many of the ensemble penalties are dependent on the number of models; for example, it is easy to see that the Max Aleatoric value could scale poorly with more models.

How Does Penalty Distribution Change with Model Count? We now vary the number of models used in the calculation of the penalties and plot their respective distributions; an illustrative example is shown in Fig. 2 with full results in App. B. The scaling of penalties relying on max over sets is most affected with increasing the number of models due to admitting more extreme values, and we observe that the distribution shape of Max Aleatoric changes significantly as we admit more models, which

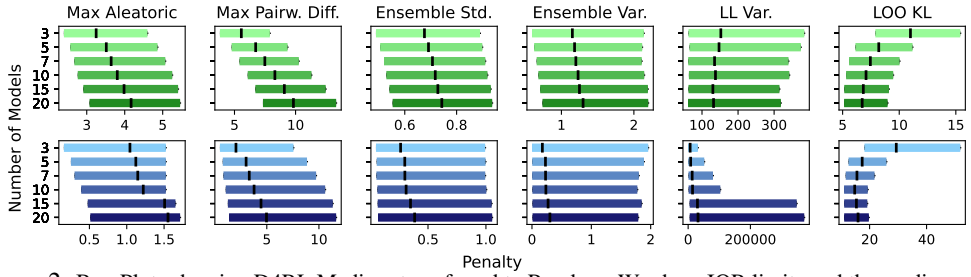


Figure 2: Box Plots showing D4RL Medium transferred to Random. We show IQR limits and the median value denoted by the black vertical line. Green = HalfCheetah, Blue = Hopper. Max Aleatoric, Max Pairw. Diff. and LOO KL are unstable w.r.t. ensemble member count. In contrast, ensemble variance and std. are far more stable.

we validate in App. C. This impacts the tuning of this hyperparameter, as we have to contend with a changing distribution along with calibration quality (which we explore in the next section). Finally, we observe that the Ensemble penalties change the least with differing model count, highlighting their ease of tuning; this is clearly a desirable property for designing such metrics going forward.

How does Penalty Performance Scale with Model Count? Empirically, there exists an optimal number of models to use in an ensemble for model-based RL (Kurutach et al., 2018; Matsushima et al., 2021). Up to now, heuristics have been used to select how many models we use for uncertainty estimation, despite it being possible to use a different number of models for dynamics prediction and uncertainty estimation. For instance, in MOPO, transitions are generated with five Elite models, but all seven models are used to calculate the penalty. In MOReL, four models are used for both transitions and penalty prediction. Therefore, we wish to understand if there is merit to using a larger number of models for uncertainty estimation compared with next state prediction. We provide a snapshot in Fig. 3, showing the aggregated results on the True Model-Based data in Hopper, with full results in App. B. We see there is no clear consensus, and that the optimal number of models is highly dependent on environment, the behavior data, and penalty type, with some settings showing improved calibration with model count and vice-versa. This clearly justifies treating the number of models as a hyperparameter that is important to tune, especially on transfer tasks. Interestingly, we observe that it is possible to simultaneously improve rank (ρ) correlation, but reduce bivariate (r) correlation, especially with the MOPO penalty. This again suggests that the number of models not only affects the quality of the estimation, but *also its distributional shape*.

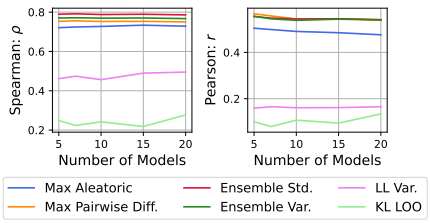


Figure 3: Plot of how error and penalty correlation changes with model number in Hopper across all datasets (i.e., Random through to Expert).

5.2 The Weight of Uncertainty λ

To weight penalty against reward, MOPO introduces a parameter λ that trades off between the two terms. In their paper, the authors sweep over $\lambda \in \{1, 5\}$ for each environment. However, the optimal values may lie outside this region. Furthermore, we have shown this value will need to drastically change to account for using a different penalty or even number of models.

5.3 The Rollout Horizon h

The horizon h of the rollouts plays a crucial role in offline RL. Longer horizon rollouts increase the likelihood of errors in the transitions (we verify this intuition in App. D), but conversely can improve performance when errors are properly managed (Janner et al., 2019; Pan et al., 2020). Furthermore, as highlighted in Fig. 1a, the model can generalize, and dynamics error does not necessarily increase with drift away from the offline dataset. Instead, we observe spikes, and note it is possible to recover from these to valid states and transitions. It is therefore imperative that a penalty captures these spikes over the course of an entire model rollout with horizon h, and down-weights the reward accordingly.

Using this observation, we design a novel experiment that treats these spikes as "positive" labels, and normalize each penalty to $[0, 1]$. This converts the penalties into a probabilistic classifier, and we evaluate how well they classify these events that occur increasingly under longer h. This is precisely the intuition behind the LOO KL and LL Var approaches, whereby the penalty acts as an

Table 2: Performance of different penalties as OOD event detectors averaged over all datasets in Hopper and HalfCheetah (i.e., Random through to Expert) showing $\pm$ 1 SD over 12 seeds. AUC is "Area Under Curve" and AP is "Average Precision". The best (highest) in each column is highlighted in **bold**.

	Percentile											
	90th				95th				99th			
	Dynamics		Distribution		Dynamics		Distribution		Dynamics		Distribution	
Penalty	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Max Aleatoric	0.89±0.01	0.50±0.02	0.76±0.01	0.35±0.01	0.89±0.00	0.35±0.02	0.80±0.01	0.27±0.01	0.92±0.00	0.20±0.04	0.89±0.03	0.16±0.04
Max Pairwise Diff.	**0.90**±0.00	0.54±0.01	0.77±0.01	0.34±0.01	**0.91**±0.00	0.40±0.02	0.81±0.01	0.28±0.01	**0.93**±0.00	0.26±0.01	0.89±0.02	0.15±0.02
Ensemble Std.	**0.90**±0.00	0.55±0.01	**0.79**±0.01	**0.38**±0.01	**0.91**±0.00	0.40±0.02	**0.83**±0.01	**0.31**±0.01	**0.93**±0.00	0.25±0.02	**0.90**±0.02	**0.18**±0.02
Ensemble Var.	**0.90**±0.00	**0.56**±0.01	0.78±0.01	0.35±0.01	**0.91**±0.00	**0.42**±0.02	0.82±0.01	0.29±0.01	**0.93**±0.00	**0.27**±0.01	0.89±0.02	0.16±0.02
LL Var.	0.66±0.03	0.33±0.00	0.74±0.02	0.33±0.00	0.67±0.02	0.21±0.02	0.76±0.02	0.25±0.02	0.73±0.03	0.09±0.01	0.81±0.02	0.11±0.01
LOO KL	0.59±0.03	0.21±0.01	0.68±0.00	0.24±0.02	0.60±0.02	0.12±0.00	0.70±0.01	0.14±0.02	0.65±0.03	0.04±0.00	0.72±0.02	0.05±0.00

anomaly detector, removing detrimental transitions that lie above a threshold. This is the regime we focus on here, where binary detection is more important than correlation. Finally, we assess two "True Model-Based" errors: the dynamics error as before, and introduce the distance from the offline distribution trained on, which we calculate as the 2-norm between a state-action tuple and its nearest point in the offline data (Dadashi et al., 2021); these are called "Dynamics" and "Distribution" respectively. We provide precision-recall curves and more details on this experiment in App. D and E.

We observe in Table 2 that the penalties are powerful at identifying dynamics discrepancy, but not as accurate at identifying when the world-model data is out-of-distribution with respect to the offline data. This is a well-known phenomenon in deep neural networks and has been recently investigated in terms of feature collapse (Van Amersfoort et al., 2020), where latent representations of points far away in the input space get mapped close together. On the other hand, this shows an important distinction between the regularization induced by MBRL uncertainty and explicit state-action regularization in model-free approaches, such as Kumar et al. (2020); Wu et al. (2021). In the latter approaches, policies are penalized for taking out of distribution actions w.r.t. the offline dataset, but this is not always the case with policies trained under MBRL and uncertainty penalties. The success of MBRL methods in RL may therefore lie in the generation of state-action samples that are **OOD but represent accurate dynamics**, thus facilitating dynamics generalization in policies; recent work has shown that augmenting dynamics improves offline RL policy generalization (Ball et al., 2021). We believe future work understanding the implications of this property is vitally important.

6 Testing the Limits of Current Approaches

Given our previous analysis, in this section we seek to answer the following question: how well can existing methods perform with a more optimal selection of the discussed hyperparameters? To answer this, we consider, first, a naïve selection of one hyperparameter set across all environments (based on our previous analysis), and then more definitively, tuning the configuration for each individual D4RL MuJoCo environment using a state-of-the-art Bayesian Optimization (BO) algorithm (Wan et al., 2021). Our first set of results show that following our analysis can provide significant gains over existing baselines, whilst the second beats the current SoTA. Note, previous analysis focused on HalfCheetah and Hopper environments, so we extend our evaluation to Walker2d as a held-out test.

General applicability of our insights. Two of our main takeaways in Sections 4 and 5 are that we should favor the canonical Ensemble penalties and longer rollout horizons. To test these claims, we design an experiment where we fix $h = 20$ for the horizon (c.f. $h = 5$ in MOPO at most), and only use Ensemble Std. as our penalty (see App. G for details). Since tuning the penalty weight λ per environment is unrealistic, we employ an automatic penalty tuning scheme, analogous to the automatic entropy tuning used in Haarnoja et al. (2018). We tune the penalty weight on-the-fly to a constraint value of $\Lambda = 1$, meaning we *use only a single hyperparameter across all environments*. Full details on the penalty weight tuning are provided in App. I. With this approach, we get an average reward of **49.0** in the D4RL locomotion test suite (Fu et al., 2021a), an increase of **43%** over MOPO, which was grid-searched per environment.

This clearly shows that applying the findings from our analysis provides large performance gains generally. This result is the best we know for a single hyperparameter setup, and is particularly significant as other offline MBRL algorithms tune many hyperparameters per environment. This 'zero-shot' hyperparameter restriction is also the most realistic application of offline RL to real world problems. If we were to allow ourselves to take the maximum over just 2 hyperparameter setups (the second setup being $h = 10$, $\Lambda = 0.5$), we achieve an average reward of **57.8**, an increase of **69%** over MOPO. We show the full results in Table 8 in App. I with improvement probabilities.

Table 3: Best hyperparameters discovered by our BO algorithm, followed by a comparative evaluation on the D4RL benchmark suite against other model-based RL algorithms. We use D4RL v0 datasets. The raw score for Optimized† and MOPO† was taken to be the average over the last 10 iterations of policy training, averaged over 4 seeds and showing ± 1 SD. Results of MOPO and COMBO were taken from the COMBO paper. Results for MOReL were taken from its paper. ⋆ indicates $p < 0.05$ for Welch's t-test for gain over MOPO. †Run on our codebase. ‡Authors' reported scores. ◦Authors used D4RL v2, which has more performant offline data.

Environment		Discovered Hyperparameters				Optimized†	MOPO†	MOPO‡	MOReL◦	COMBO
		N	λ	h	Penalty					
HalfCheetah	random	10	6.64	12	Ensemble Std	31.7 ±1.5	32.7 ±1.7	35.4	25.6	**38.8**
	mixed	11	0.96	37	Ensemble Var	**58.0** ±2.5	52.8 ±1.1	53.1	40.2	55.1
	medium	12	5.92	6	Ensemble Var	45.7 ±2.6	46.5 ±0.7	42.3	42.1	**54.2**
	med.-exp.	7	4.56	5	Max Aleatoric	**104.2** ±5.7 ⋆	67.6 ±23.6	63.3	53.3	90.0
Hopper	random	6	4.46	47	Ensemble Std	12.1 ±0.2 ⋆	4.2 ±1.5	11.7	**53.6**	17.8
	mixed	7	5.90	5	Max Aleatoric	90.8 ±11.1 ⋆	66.7 ±27.8	67.5	**93.6**	73.1
	medium	7	37.28	42	Ensemble Std	69.3 ±15.2 ⋆	17.3 ±6.3	28.0	**95.4**	94.9
	med.-exp.	12	39.08	43	Max Aleatoric	105.8 ±1.2 ⋆	24.9 ±5.5	23.7	108.7	**111.1**
Walker2d	random	10	0.21	12	Ensemble Var	21.7 ±0.1 ⋆	13.6 ±1.4	13.6	**37.3**	7.0
	mixed	13	2.48	47	Ensemble Std	**65.8** ±17.4 ⋆	37.6 ±20.6	39.0	49.8	56.0
	medium	8	5.28	14	Ensemble Std	**79.7** ±2.3 ⋆	-0.1 ±0.0	17.8	77.8	75.5
	med.-exp.	12	0.99	37	Ensemble Std	**97.1** ±4.9 ⋆	46.2 ±27.0	44.6	95.6	96.1
Average Score		-	-	-	-	**65.2** ±5.4 ⋆	34.2 ±9.8	36.7	64.4	64.1

Testing the limits of current approaches. Next, we wish to further validate that our earlier theoretical analysis can correspond to strong empirical performance gains by performing BO over the key hyperparameters. Details on the BO algorithm are listed in App. G. We define our search space over hyperparameters most related to *uncertainty quantification*:

- **Penalty type (categorical):** taking values over {Max Aleatoric, Max Pairwise Diff, LOO KL, LL Var, Ensemble Std, Ensemble Variance}.
- **Penalty scale λ (continuous):** taking values over $[1, 100]$.
- **h (integer):** taking values over $\{1, 2, \ldots, 50\}$.
- **Models N (integer):** taking values over $\{1, 2, \ldots, 15\}$.

Table 3 shows the optimal hyperparameters under BO. We note that Ensemble penalties are mainly selected, corroborating the findings in our analysis that these are most correlated with model error. We observe that Max Pairwise Diff is not chosen, likely because ensemble penalties are better correlated with true dynamics error, and are more stable under tuning since their scaling changes less with model number; we know that Max Pairwise Diff has very similar shape statistics to Ensemble Std. (App. C). Finally, we also observe these solutions have lower performance variance than MOPO.

The selection of Max Aleatoric is also explainable; we observe it displays significantly lower skew and kurtosis than all other metrics (App. C), while still maintaining strong rank correlation. We also found that in all Hopper experiments, Ensemble Var. never achieved high performance, despite the only difference with Ensemble Std. being its distributional shape. Interestingly, in HalfCheetah, the opposite is true, with Ensemble Var. delivering significant performance gains. This implies that distributional shape may play as important a role as calibration, and advocates the learning of *meta-parameters* that control for this. Finally, in Walker2d, the well-grounded ensemble penalties win in all cases. We note that values of the rollout horizon h and penalty weight λ differ greatly from those chosen in the original MOPO paper, which chooses both from $\{1, 5\}$. Notably, the Hopper and Walker2d environments can prefer a much longer rollout length and higher penalty weight, even accounting for penalty magnitudes. Again this is backed up by our analysis; along a single rollout, dynamics errors do not necessarily accumulate, they simply become more likely to occur. Therefore, as long as we penalize errors appropriately, we can handle longer rollouts and, as a result, generate more on-policy data. The number of models used to compute the uncertainty estimates can also differ greatly from the standard 7. This again aligns with our findings that using more models for uncertainty estimation can be beneficial, but is dependent on environment, data, and penalty.

Table 3 also demonstrates how these unconventional hyperparameter choices fare against state-of-the-art offline MBRL algorithms. We spent considerable effort ensuring that our implementation of MOPO matched the authors' results using the same hyperparameters. We note the two are very similar[1], thereby allowing us to make a faithful comparison when modifying hyperparameters. Our approach, labeled "Optimized†", achieves *statistically significant* improvements over MOPO on 9

[1]There was a disparity in Walker2d-medium, but this was also noted in Ball et al. (2021)

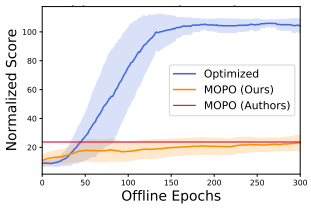


Figure 4: MOPO performance on the Hopper medium-expert environment.

Environment		MOPO†	Optimized†	CQL
pen	cloned	5.4 ±10.8	23.0 ±4.2	39.2
	human	6.2 ±7.8	19.0 ±7.9	37.5
	expert	15.1 ±9.7	50.6 ±10.5	107.0
hammer	cloned	0.2 ±0.1	5.2 ±1.5	2.1
	human	0.2 ±0.0	0.5 ±0.8	4.4
	expert	6.2 ±8.4	23.3 ±4.1	86.7

Table 4: Comparative evaluation on the D4RL Adroit v0 dataset against Model Free CQL

out of 12 environments, validating our prior analysis over the key design choices. As an additional bonus, and it is not the stated aim of this work, our approach achieves state-of-the-art performance on five HalfCheetah and Walker2d environments by a considerable margin. Further notable results include the Hopper mixed and Hopper medium-expert environments, in which we show we are able to tune the MOPO-like method up to the performance of COMBO (Yu et al., 2021) and MOReL. The importance of good uncertainty estimation and hyperparameter selection is shown visually in Fig. 4 where we improve MOPO performance by over 5× whilst obtaining a stable solution.

As aforementioned, we found our policies are more stable than previous works and consequently *do not need to cherry-pick* high performing checkpoints[2]. Instead, we report the average performance over our final 10 policy-improvement iterations. It should be noted that stability during training (Chan et al., 2020) is paramount for successful policy deployment in offline RL, and we should therefore prioritize hyperparameters that ensure this. We further confirm the reliability of our evaluation using the `rliable` framework (Agarwal et al., 2021) in Fig. 1b, showing that the improvement over MOPO (with 95% bootstrap CIs shaded) is clear in both MuJoCo and Adroit.

Results on Adroit dexterous hand manipulation tasks. We present results in Table 4 on the Adroit Pen and Hammer environments which, as far as we are aware, *have not previously been used in offline MBRL*, and present very different challenges to the locomotion tasks. These tasks feature sparse rewards, real human demonstrations and narrow data distributions. We compare against the current state-of-the-art *model-free* algorithm (CQL, Kumar et al. (2020)) and find that offline MBRL can learn useful policies in the Adroit domains, providing the best performance seen so far on the hammer-cloned setting. Best found penalties and hyperparameters are listed in App. J, and mirror the findings in the locomotion experiments. We believe issues with the world model not accurately capturing sparse rewards may account for any major performance difference. Our work is therefore an important step towards bridging the gap between model-based and model-free methods for sparse reward tasks, especially in the offline setting where exploration is not possible. We define MOPO to be the best performance with the Max Aleatoric penalty, searching λ, h in $\{1, 5\}^2$.

7 Conclusion

In this paper, we rigorously evaluated the impact of various key design choices on offline MBRL, comparing for the first time a number of different uncertainty penalties used in the literature. By proposing novel evaluation protocols, we have also gained key insights into the nature of uncertainty in offline MBRL that we believe benefits the RL community. We demonstrated the impact of this analysis by significantly improving upon existing offline MBRL by using vastly different key hyperparameters, obtaining statistically significant performance improvements in almost all benchmarks.

Going forward, we are excited by developments in offline evaluation (Chen et al., 2021; Fu et al., 2021b) to accurately assess agent performance without querying the environment. This would open the door for population-based training methods (Jaderberg et al., 2017; Parker-Holder et al., 2020), which have shown great success in online MBRL (Zhang et al., 2021). Furthermore, throughout the paper we have highlighted potential areas of interest, from better understanding the generalization provided by world models, through to the development of meta-parameters controlling penalty distribution shape. We also highlight key issues in implementation in App. F as we strongly believe this is a vital frontier for disentangling the effect of algorithmic innovations from code-level details. Finally, Offline MBRL so far has only focused on deterministic environments; the ensemble penalties we investigate support the modeling of stochastic dynamics, and the novel tools for analysis we develop here can be readily applied to such settings.

[2]It is unclear what procedure is used in some prior work (indeed issues have been raised about this).

ACKNOWLEDGMENTS

The authors would like to acknowledge Rishabh Agarwal for helpful feedback during the project. We would also like to thank the anonymous reviewers for their constructive feedback, which helped to improve the paper. Cong Lu is funded by the Engineering and Physical Sciences Research Council (EPSRC). Philip J. Ball is funded through the Willowgrove Studentship.

REFERENCES

Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective Dyna-style planning under limited model capacity. In *ICML*, pp. 1–10, 2020. URL http://proceedings.mlr.press/v119/abbas20a.html.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34. 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nIAxjsniDzg.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*. 2020.

Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 619–629. PMLR, 18–24 Jul 2021. URL http://proceedings.mlr.press/v139/ball21a.html.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/blundell15.html.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*. 07 2018.

Stephanie C.Y. Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJlpYJBKvH.

Yutian Chen, Liyuan Xu, Caglar Gulcehre, Tom Le Paine, Arthur Gretton, Nando de Freitas, and Arnaud Doucet. On instrumental variable regression for deep offline policy evaluation, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31*, pp. 4754–4765. 2018.

Alexander Cowen-Rivers, Daniel Palenicek, Vincent Moens, Mohammed Abdullah, Aivar Sootla, Jun Wang, and Haitham Bou Ammar. Samba: safe model-based & active reinforcement learning. *Machine Learning*, pp. 1–31, 01 2022. doi: 10.1007/s10994-021-06103-6.

Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal Wasserstein imitation learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=TtYSU29zgR.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 465–472, 2011.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pp. 1–50, 2021.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2020.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. URL http://jmlr.org/papers/v6/ernst05a.html.

Angelos Filos, Sebastian Farquhar, Aidan N. Gomez, Tim G. J. Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4{RL}: Datasets for deep data-driven reinforcement learning, 2021a.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, ziyu wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Thomas Paine. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=kWSeGEeHvF8.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/guo17a.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*. 2019.

Sham M Kakade. A natural policy gradient. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. URL https://proceedings.neurips.cc/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL : Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*. 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning, 2021.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2796–2804. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/kuleshov18a.html.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*. 2020.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

David John Cameron Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, USA, 1992. UMI Order No. GAX92-32200.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/118921efba23fc329e6560b27861f0c2-Paper.pdf.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3hGNqpI4WS.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015.

D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pp. 55–60 vol.1, 1994.

Muhammad Omer, Rami Ahmed, Benjamin Rosman, and Sharief F. Babikir. Model predictive-actor critic reinforcement learning for dexterous manipulation. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1–6, 2021. doi: 10.1109/ICCCEEE49695.2021.9429677.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/8558cb408c1d76621371888657d2eb1d-Paper.pdf.

Aldo Pacchiano, Philip Ball, Jack Parker-Holder, Krzysztof Choromanski, and Stephen Roberts. Towards tractable optimism in model-based reinforcement learning. In *Uncertainty in Artificial Intelligence*. 2021.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Çaglar Gülçehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *CoRR*, abs/2007.09055, 2020. URL https://arxiv.org/abs/2007.09055.

Feiyang Pan, Jia He, Dandan Tu, and Qing He. Trust the model when it is confident: Masked model-based actor-critic. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10537–10546. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/77133be2e96a577bd4794928976d2ae2-Paper.pdf.

Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17200–17211. Curran Associates, Inc., 2020.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. MBRL-Lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*, 2020.

Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pp. 8276–8285. PMLR, 2020.

Gabriele Scalia, Colin A Grambow, Barbara Pernici, Yi-Pei Li, and William H Green. Evaluating scalable uncertainty estimation methods for deep learning-based molecular property prediction. *Journal of chemical information and modeling*, 60(6):2697–2717, 2020.

Jian Shen, Han Zhao, Weinan Zhang, and Yong Yu. Model-based policy optimization with unsupervised model adaptation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2823–2834. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1dc3a89d0d440ba31729b0ba74b93a33-Paper.pdf.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9690–9700. PMLR, 13–18 Jul 2020.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10663–10674. PMLR, 18–24 Jul 2021.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. In *To Appear: The International Conference on Learning Representations (ICLR)*. 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*. 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2021.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Limitations and Future Work

One limitation of our analysis is that properly evaluating design choices and the downstream performance of particular hyperparameter choices requires evaluating the final policy online with a small number of samples. While this is acceptable for our analysis, fully offline evaluation (Chen et al., 2021c; Fu et al., 2021), would open the door to fully realizing these benefits in general. As in online reinforcement learning, one could obtain superior per-environment configurations that could vastly improve on the baseline. Going even further, this could also open the door to population-based training methods (Jaderberg et al., 2017; Wan et al., 2022), which have shown great success in online MBRL (Zhang et al., 2021b). These methods dynamically adapt the hyperparameters of a diverse population of agents by preferring those that lead to higher evaluated return. Nonetheless, we are encouraged that Sun et al. (2023) showed that the insights from our paper are more generally applicable and showed that a variant of MOPO with our uncertainty quantification is still close to SOTA on newer datasets.

Next, a further interesting direction for future work is understanding the role of penalty distribution shape. While we showed in our analysis that the ensemble penalties are better correlated with true dynamics error, the standard deviation was preferred for Hopper, whereas the variance was preferred for HalfCheetah. As one is simply the square of the other, there is no difference in correlation but simply with the distribution of values. This problem is related to the general problem of reward shaping (Ng et al., 1999) in RL and a more general approach could be to start with either the standard deviation or variance and consider general families of monotonic functions to shape the distribution of its values. For example, the function $f(x) = 1 - (1 - x^n)^{\frac{1}{n}}$ maps the range $[0, 1]$ to itself monotonically with a controllable degree of sharpness given by $n > 0$. This could lead to a unified penalization scheme that could also encompass the use of max operators as in MOPO and MOReL that tend to skew the distributions towards higher values.

Finally, as we alluded to in the Chapter 5, further research in model class could lead to entirely replacing deep ensembles with better forms of uncertainty quantification. Sims et al. (2023) also studies limitations of the current prevailing paradigm of model uncertainty-based penalization. As we look to the next chapter, the methods we develop here will also aid in the development of offline model-based methods with visual observations.

7

Challenges and Opportunities in Offline Reinforcement Learning from Visual Observations.

In our final chapter, we look toward developing algorithms and benchmarks for offline reinforcement learning from visual observations. While offline reinforcement learning has shown great promise in proprioceptive settings (Ernst et al., 2005; Levine et al., 2020), applying these approaches to visual observations is considerably less well understood. Developing such algorithms could enable reinforcement learning to be more widely applicable to real-world settings, where we have access to vast quantities of visual observations of desirable behaviors. For example, in autonomous driving (Kendall et al., 2018), large quantities of such data already exist but have not been fully utilized (Maddern et al., 2016; Yu et al., 2020a). While some progress has been made in this area (Chen et al., 2021a; Florence et al., 2021; Rafailov et al., 2021; Shah et al., 2021), studies have often focused on disparate tasks with bespoke datasets. In particular, conspicuously missing is a publicly available and comprehensive benchmarking suite with carefully evaluated baselines. With this work, we aim to address both of these needs.

Following the proprioceptive setting, we begin by establishing model-based and model-free offline baselines by making simple adjustments to two popular online algorithms, DreamerV2 (Hafner et al., 2020b) and DrQ-v2 (Yarats et al., 2021). As discussed in Section 2.2.2, DreamerV2 is a model-based algorithm that predicts dynamics in latent space. This naturally allows us to construct a pessimistic MDP as in Chapter 6 by considering the mean-disagreement of an ensemble of RSSMs.[1] We call this algorithm **Offline DV2**. On the other hand, we note that the base policy optimizer in DrQ-v2 shares similarities with TD3 (Fujimoto et al., 2018), which as discussed in Section 2.3.2, can be extended to the offline setting by adding a regularizing behavioral-cloning term to the policy loss. Applying the same logic to DrQ-v2 leads us to our second algorithm, **DrQ+BC**.

Next, we present a benchmark for offline RL from visual observations of DMControl Suite (DMC) tasks (Tassa et al., 2020) which include counterparts to the standard proprioceptive MuJoCo environments. Our benchmark, **Vision Datasets for Deep Data-Driven RL** (V-D4RL), follows the design principles of the popular D4RL benchmark (Fu et al., 2020), and is the first publicly available benchmark for continuous control which features a wide variety of behavioral policies. This allows us to evaluate the relative strengths and weaknesses of our model-based and model-free baselines.

Finally, we identify *three key desiderata* for realistic offline RL from visual observations: robustness to distractions (Stone et al., 2021)[2], generalization across dynamics (Zhang et al., 2021a), and improved performance at scale. We present a suite of evaluation protocols and additional datasets to V-D4RL designed to test whether offline RL algorithms satisfy these desiderata. This allows us to establish baselines with our algorithms and elucidate challenges for future work.

[1]This represents the epistemic component of the ensemble variance we introduced in the previous chapter, we discuss this choice in Appendix D.4 of the paper.

[2]A standard setting from the online literature that tests robustness to visual distractions that do not affect the underlying dynamics.

Cong Lu*, Philip J. Ball*, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and Opportunities in Offline Reinforcement Learning from Visual Observations. **In TMLR, 2023.**

Challenges and Opportunities in Offline Reinforcement Learning from Visual Observations

Cong Lu* *cong.lu@stats.ox.ac.uk*
University of Oxford

Philip J. Ball* *ball@robots.ox.ac.uk*
University of Oxford

Tim G. J. Rudner *tim.rudner@cs.ox.ac.uk*
University of Oxford

Jack Parker-Holder *jackph@robots.ox.ac.uk*
University of Oxford

Michael A. Osborne *mosb@robots.ox.ac.uk*
University of Oxford

Yee Whye Teh *y.w.teh@stats.ox.ac.uk*
University of Oxford

Reviewed on OpenReview: *https://openreview.net/forum?id=1QqIfGZOWu*

Abstract

Offline reinforcement learning has shown great promise in leveraging large pre-collected datasets for policy learning, allowing agents to forgo often-expensive online data collection. However, offline reinforcement learning from *visual observations* with continuous action spaces remains under-explored, with a limited understanding of the key challenges in this complex domain. In this paper, we establish simple baselines for continuous control in the visual domain and introduce a suite of benchmarking tasks for offline reinforcement learning from visual observations designed to better represent the data distributions present in real-world offline RL problems and guided by a set of desiderata for offline RL from visual observations, including robustness to visual distractions and visually identifiable changes in dynamics. Using this suite of benchmarking tasks, we show that simple modifications to two popular vision-based online reinforcement learning algorithms, DreamerV2 and DrQ-v2, suffice to outperform existing offline RL methods and establish competitive baselines for continuous control in the visual domain. We rigorously evaluate these algorithms and perform an empirical evaluation of the differences between state-of-the-art model-based and model-free offline RL methods for continuous control from visual observations. All code and data used in this evaluation are open-sourced to facilitate progress in this domain.

Open-sourced code and data for the V-D4RL benchmarking suite are available at:
https://github.com/conglu1997/v-d4rl.

*Equal contribution. Correspondence to cong.lu@stats.ox.ac.uk and ball@robots.ox.ac.uk.

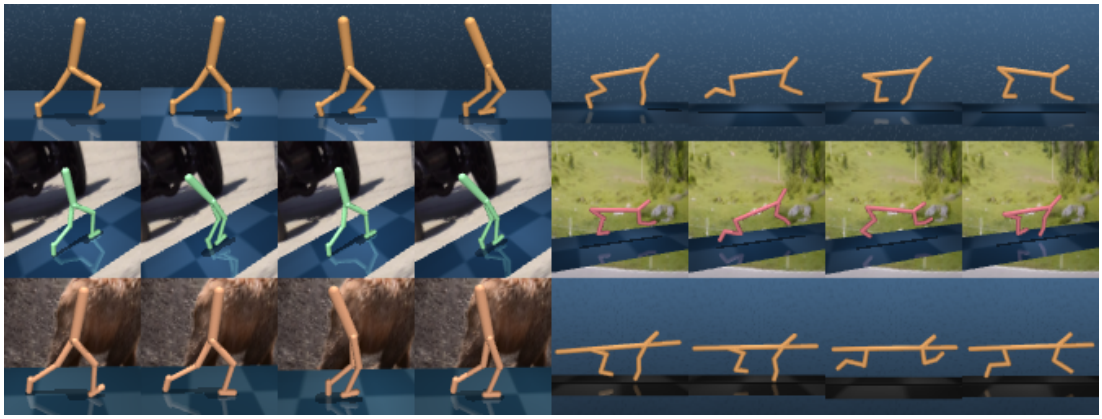


Figure 1: V-D4RL is a benchmarking suite for offline reinforcement learning from visual observations based on the DMControl Suite (Tassa et al., 2020), which includes a comprehensive set of D4RL-style datasets and modalities unique to learning from visual observations.

1 Introduction

The reinforcement learning (RL, Sutton & Barto (1992)) paradigm is conventionally characterized as learning from *online interaction* with an environment. However, in many real-world settings, such as robotics or clinical decision-making, online interactions can be expensive or impossible to perform due to physical or safety constraints. *Offline* (or batch) reinforcement learning (Ernst et al., 2005; Levine et al., 2020) aims to address this issue by leveraging pre-collected datasets to train and deploy autonomous agents without requiring online interaction with an environment.

While offline reinforcement learning algorithms, both model-based (Yu et al., 2020b; Kidambi et al., 2020; Argenson & Dulac-Arnold, 2021) and model-free (Kumar et al., 2020; Kostrikov et al., 2021; Fujimoto & Gu, 2021; Wu et al., 2019; Fujimoto et al., 2019; Kumar et al., 2019), have mastered challenging continuous control tasks, most prior works have relied on access to proprioceptive states focusing on standardized benchmarking suites (Fu et al., 2021). In contrast, studies of offline reinforcement learning from *visual observations* (Rafailov et al., 2021; Shah et al., 2021; Chen et al., 2021; Florence et al., 2021) have often focused on disparate tasks with bespoke datasets due to the lack of a well-designed benchmarking suite with carefully evaluated baselines. With this work, we aim to address both of these needs.

Training agents from visual observations offline provides an opportunity to make reinforcement learning more widely applicable to real-world settings, where we have access to vast quantities of visual observations of desirable behaviors. For example, in autonomous driving (Kendall et al., 2018), large quantities of visual offline data already exist but have not been fully utilized (Maddern et al., 2016; Yu et al., 2020a). Similarly, in robotics, data collection is expensive due to costs associated with set-up and human supervision. Effective, transferable offline reinforcement learning could allow us to reuse datasets gathered previously for different tasks or settings for unseen new problems (Chebotar et al., 2021). Unlocking this potential would represent significant progress towards learning general-purpose agents for realistic problems.

To enable the development of effective, robust, and adaptive algorithms for offline RL from visual observations, we present a suite of carefully designed datasets and benchmarking tasks for this burgeoning domain. We use these tasks to establish simple performance baselines, to study how the composition of vision-based offline datasets affects the performance of different types of RL algorithms, and to evaluate the extent to which algorithms for offline RL from visual observations satisfy a set of *desiderata*, including robustness to visual distractions, generalization across environment dynamics, and improved performance at scale. Our evaluation identifies clear failure modes of the baseline methods and highlights opportunities and open problems for future work which can be tackled with our benchmark.

Recent progress in offline reinforcement learning from proprioceptive observations has been driven by well-designed and easy-to-use evaluation testbeds and baselines. We hope that V-D4RL and the analysis in this paper will help facilitate the development of robust RL agents that leverage large, diverse, and often imperfect offline datasets of visual observations across tasks and deployment settings.

The core contributions of this paper are as follows:

1. We present a benchmark for offline RL from visual observations of DMControl Suite (DMC) tasks (Tassa et al., 2020). This benchmark, **Vision Datasets for Deep Data-Driven RL** (V-D4RL), follows the design principles of the popular D4RL benchmark (Fu et al., 2021), and to our knowledge is the first publicly available benchmark for continuous control which features a wide variety of behavioral policies.

2. We identify **three key desiderata** for realistic offline RL from visual observations: robustness to distractions (Stone et al., 2021), generalization across dynamics (Zhang et al., 2021), and improved performance for offline reinforcement learning at scale. We present a suite of evaluation protocols designed to test whether offline RL algorithms satisfy these desiderata.

3. We establish model-based and model-free baselines for offline RL from visual observations. We do so by modifying two popular online RL algorithms, **DreamerV2** (Hafner et al., 2020b) and **DrQ-v2** (Yarats et al., 2021a), which showcase the relative strengths of model-based and model-free algorithms for offline RL from visual observations. We use these algorithms to provide simple baselines for the aforementioned desiderata to serve as a measure of progress for future advances in this domain.

2 Preliminaries

We model the environment as a Markov Decision Process (MDP), defined as a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(s'|s, a)$ the transition dynamics, $R(s, a)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in (0, 1)$ the discount factor. The standard goal in online reinforcement learning is to optimize a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$ through interactions with the environment.

In *offline reinforcement learning*, the policy is not deployed in the environment until test time. Instead, the algorithm has access to a fixed dataset $\mathcal{D}_{\text{env}} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{N}$, collected by one or more behavioral policies π_b. Following Yu et al. (2020b), we refer to the distribution from which $\mathcal{D}_{\text{env}}$ was sampled as the *behavioral distribution*.

We first describe recent advancements in offline RL and RL from visual observations through the lens of model-based and model-free methods.

2.1 Offline Reinforcement Learning Paradigms

Model-based. A central problem in offline reinforcement learning is over-estimation of the value function (Sutton & Barto, 1992) due to incomplete data (Kumar et al., 2019). Model-based methods in offline RL provide a natural solution to this problem by penalizing the reward from model rollouts by a suitable measure of uncertainty. Yu et al. (2020b) provide a theoretical justification for this approach by constructing a pessimistic MDP (P-MDP) and lower-bounding the expected true return, $\eta_M(\pi)$, using the error between the estimated and true model dynamics. Since this quantity is usually not available without access to the true environment dynamics, algorithms such as MOPO and MOReL (Yu et al., 2020b; Kidambi et al., 2020) penalize reward with a surrogate measure of uncertainty. These algorithms train an ensemble of K probabilistic dynamics models (Nix & Weigend, 1994) and define a heuristic based on the ensemble predictions. Recent work (Lu et al., 2022) has shown that a better approach to approximating the true dynamics error is to use the standard deviation of the ensemble's mixture distribution instead, as proposed by Lakshminarayanan et al. (2017).

Model-free. In the model-free paradigm, we lose the natural measure of uncertainty provided by the model. In lieu of this, algorithms such as CQL (Kumar et al., 2020) attempt to avoid catastrophic overestimation by penalizing actions outside the support of the offline dataset with a wide sampling distribution over the action bounds. Recently, Fujimoto & Gu (2021) have shown that a minimal approach to offline reinforcement learning works in proprioceptive settings, where offline policy learning with TD3 (Fujimoto et al., 2018) can be stabilized by augmenting the loss with a behavioral cloning term.

2.2 Reinforcement Learning from Visual Observations

Recent advances in reinforcement learning from visual observations have been driven by use of data augmentation, contrastive learning and learning recurrent models of the environment. We describe the current dominant paradigms in model-based and model-free methods below.

Model-based (DreamerV2, Hafner et al. (2020b)). DreamerV2 learns a model of the environment using a Recurrent State Space Model (RSSM, Hafner et al. (2019; 2020a), and predicts ahead using compact model latent states. The particular instantiation used in DreamerV2 uses model states s_t containing a deterministic component h_t, implemented as the recurrent state of a Gated Recurrent Unit (GRU, Chung et al. (2014)), and a stochastic component z_t with a categorical distribution. The actor and critic are trained from imagined trajectories of latent states, starting at encoded states of previously encountered sequences.

Model-free (DrQ-v2, Yarats et al. (2021b)). DrQ-v2 is an off-policy algorithm for vision-based continuous control, which uses data-augmentation (Laskin et al., 2020; Yarats et al., 2021c) of the state and next state observations. The base policy optimizer is DDPG (Lillicrap et al., 2016), and the algorithm uses a convolutional neural network (CNN) encoder to learn a low-dimensional feature representation.

3 Related Work

There has been significant progress in offline RL, accompanied and facilitated by the creation of several widely used benchmarking suites. We list several here; to our knowledge, no contemporary datasets are both publicly available and feature the same range of behaviors as D4RL, nor do they feature tasks related to distractions or changed dynamics.

Benchmarks for continuous control on states. D4RL (Fu et al., 2021) is the most prominent benchmark for continuous control with proprioceptive states. The large variety of data distributions has allowed for comprehensive benchmarking (Kumar et al., 2020; Kidambi et al., 2020; Yu et al., 2020b; 2021; Kostrikov et al., 2021) and understanding the strengths and weaknesses of offline algorithms. Our work aims to establish a similar benchmark for tasks with visual observations. RL Unplugged (Gulcehre et al., 2020) also provides public visual data on DMControl tasks but only provides mixed data which is filtered on the harder locomotion suite. COMBO (Yu et al., 2021) also provides results on the DMC walker-walk environment however does not benchmark over a full set of behavioral policies and the datasets are not publicly available at the time of writing.

Analysis on characteristics of offline datasets. Recent work (Florence et al., 2021) has sought to understand when offline RL algorithms outperform behavioral cloning in the proprioceptive setting. Kumar et al. (2021) recommend BC for many settings but showed theoretically that offline RL was preferable in settings combining expert and suboptimal data. This finding is corroborated by our analysis (see Tables 1 and 4).

Vision-based discrete control datasets. Whilst there has been a lack of suitable benchmarks for vision-based offline continuous control, vision-based datasets for discrete control have been created for Atari (Agarwal et al., 2020). However, at 50M samples per environment, this benchmarking suite can be prohibitive in terms of its computational hardware requirements (see `https://github.com/google-research/batch_rl/issues/10`). We believe that V-D4RL's 100,000-sample benchmark represents a more achievable task for practitioners.

Offline vision-based robotics. Learning control offline from visual observations is an active area of robotics research closely-related to the problems considered in V-D4RL. Interesting pre-collected datasets exist in this domain but with a primary focus on final performance rather than comprehensive benchmarking on different data modalities; we list some of these here. In QT-Opt (Kalashnikov et al., 2018), data is initially gathered with a scripted exploration policy and then re-collected with progressively more successful fine-tuned robots. Chebotar et al. (2021) follow a similar setup, and explores representation learning for controllers. Mandlekar et al. (2021); Cabi et al. (2020) consider learning from combined sets of human demonstrations, data from scripted policies and random data.

4 Baselines for Offline Reinforcement Learning from Visual Observations

In this section, we begin by motivating our creation of a new benchmark (V-D4RL), introduce our new simple baselines combining recent advances in offline RL and vision-based online RL, and present a comparative evaluation of current methods on V-D4RL. Comprehensive and rigorous benchmarks are crucial to progress in nascent fields. To our knowledge, the only prior work that trains vision-based offline RL agents on continuous control tasks is LOMPO (Rafailov et al., 2021). We analyze their datasets in Section 4.3.1 and find they do not conform to standard D4RL convention.

4.1 Adopting D4RL Design Principles

In this section, we outline how to generate D4RL-like vision-based datasets for V-D4RL. To generate offline datasets of visual observations, we consider the following three DMControl Suite (DMC) environments (these environments are easy, medium and hard respectively (Yarats et al., 2021a)):

- **walker-walk**: a planar walker is rewarded for being upright and staying close to a target velocity.
- **cheetah-run**: a planar biped agent is rewarded linearly proportional to its forward velocity.
- **humanoid-walk**: a 21-jointed humanoid is rewarded for staying close to a target velocity. Due to the huge range of motion styles possible, this environment is *extremely challenging* with many local minima and is included as a stretch goal.

From these environments, we follow a D4RL-style procedure in considering five different behavioral policies for gathering the data. As in D4RL, the base policy used to gather the data is Soft Actor–Critic (SAC, Haarnoja et al. (2018)) on the proprioceptive states. We consider the following five settings:

- **random**: Uniform samples from the action space.
- **medium-replay (mixed)**: The initial segment of the replay buffer until the SAC agent reaches medium-level performance.
- **medium**: Rollouts of a fixed medium-performance policy.
- **expert**: Rollouts of a fixed expert-level policy.
- **medium-expert (medexp)**: Concatenation of medium and expert datasets above.

We provide precise specifications for medium, mixed and expert in Section 4.1.1 and discuss the choice of offline behavioral policy in Section 4.1.2. By default, each dataset consists of 100,000 total transitions (often 10× less than in D4RL) in order to respect the memory demands of vision-based tasks. The cheetah and humanoid medium-replay datasets consist of 200,000 and 600,000 transitions respectively due to the increased number of samples required to train policies on these environments. Full statistics of each dataset are given in Appendix A. In Section 5, we further extend the original V-D4RL datasets to study problem settings with multiple tasks and visual distractions as illustrated in Figure 1.

4.1.1 Data Generation Policy

To create the offline medium and expert datasets, we first train SAC (Haarnoja et al., 2018) policies on the proprioceptive states until convergence, taking checkpoints every 10,000 frames of interaction. We use a frame skip of 2, the default in other state-of-the-art vision RL algorithms (Hafner et al., 2020b; Yarats et al., 2021a). Medium policies are defined as the first saved agent during training that is able to consistently

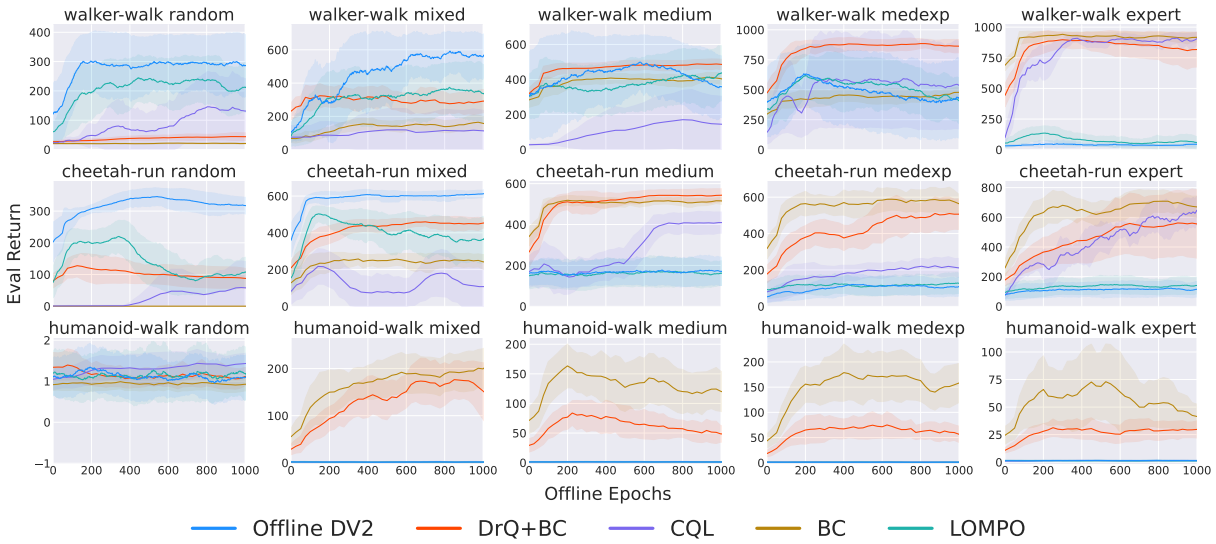


Figure 2: Rigorous comparison on the V-D4RL benchmark, each setting is averaged over six random seeds with error bar showing one standard deviation. Total gradient steps are normalized under epochs, and we plot the un-normalized evaluated return. We note that the model-free and BC baselines are far more stable than the model-based.

achieve above 500 reward in the environment. Expert policies are defined as those that have converged in the limit. For DMControl tasks, this typically means near 1,000 reward on the environment they were trained on. We confirm these thresholds are reasonable, as we observe a noticeable gap between the behavior of the medium and expert-level policies.

In order to generate the offline visual observations, we deploy the proprioceptive agents in the environment, and save the visual observations rendered from the simulator instead of the proprioceptive state. This provides us with the flexibility to *generate observations of any size* without having to retrain for that resolution (e.g., 84×84 or 64×64). As was done in D4RL, we generate data using a stochastic actor, which involves sampling actions from the Gaussian conditional distribution of the SAC policy, featuring a parameterized variance head that determines the amount of action stochasticity at each state.

For the mixed datasets, we simply store the replay buffer of the medium agent when it finishes training, and convert the proprioceptive observations into visual observations.

4.1.2 Choice of Offline Behavioral Policy

We choose proprioceptive SAC as our behavioral policy π_b mirroring Fu et al. (2021); Rafailov et al. (2021). We found that using online DrQ-v2 as the offline behavioral policy made the tasks significantly easier to learn for all agents. This suggests that the proprioceptive agent may learn behavior modes that are less biased towards being easy under vision-based methods; for instance, DrQ-v2 may be biased towards behavior modes that induce fewer visual occlusions compared to proprioceptive SAC.

4.2 Baselines

We show that for the two state-of-the-art online vision-based RL algorithms described in Section 2.2, simple adjustments from the proprioceptive literature suffice to transfer them to the offline setting. In Section 4.3, we demonstrate that these baselines provide a new frontier on our benchmark and on prior datasets. Additional details and hyperparameters for our algorithms are given in Appendix B.

Model-based. For DreamerV2, we follow Sekar et al. (2020) in constructing a bootstrap ensemble for the dynamics model, which allows us to naturally define a penalty for the reward based on the dynamics

Table 1: Final performance on V-D4RL averaged over six random seeds, with one standard deviation error. Evaluated return is mapped from $[0, 1000]$ to $[0, 100]$. Our model-based method does best on the diverse low-reward datasets, model-free on the diverse high-reward datasets, and behavioral cloning on the narrow expert data. Furthermore, we provide the mean return of the dataset for reference; full statistics are included in Appendix A.

Environment		Offline DV2	DrQ+BC	CQL	BC	LOMPO	Data Mean
walker-walk	random	**28.7 ±13.0**	5.5 ±0.9	14.4 ±12.4	2.0 ±0.2	**21.9 ±8.1**	4.2
	mixed	**56.5 ±18.1**	28.7 ±6.9	11.4 ±12.4	16.5 ±4.3	34.7 ±19.7	14.5
	medium	34.1 ±19.7	**46.8 ±2.3**	14.8 ±16.1	40.9 ±3.1	43.4 ±11.1	44.0
	medexp	43.9 ±34.4	**86.4 ±5.6**	56.4 ±38.4	47.7 ±3.9	39.2 ±19.5	70.4
	expert	4.8 ±0.6	68.4 ±7.5	**89.6 ±6.0**	**91.5 ±3.9**	5.3 ±7.7	97.0
cheetah-run	random	**31.7 ±2.7**	5.8 ±0.6	5.9 ±8.4	0.0 ±0.0	11.4 ±5.1	0.7
	mixed	**61.6 ±1.0**	44.8 ±3.6	10.7 ±12.8	25.0 ±3.6	36.3 ±13.6	19.1
	medium	17.2 ±3.5	**53.0 ±3.0**	40.9 ±5.1	**51.6 ±1.4**	16.4 ±8.3	52.4
	medexp	10.4 ±3.5	50.6 ±8.2	20.9 ±5.5	**57.5 ±6.3**	11.9 ±1.9	70.7
	expert	10.9 ±3.2	34.5 ±8.3	**61.5 ±4.3**	**67.4 ±6.8**	14.0 ±3.8	89.1
humanoid-walk	random	0.1 ±0.0	0.1 ±0.0	0.2 ±0.1	0.1 ±0.0	0.1 ±0.0	0.1
	mixed	0.2 ±0.1	**15.9 ±3.8**	0.1 ±0.0	**18.8 ±4.2**	0.2 ±0.0	27.6
	medium	0.2 ±0.1	6.2 ±2.4	0.1 ±0.0	**13.5 ±4.1**	0.1 ±0.0	57.3
	medexp	0.1 ±0.0	7.0 ±2.3	0.1 ±0.0	**17.2 ±4.7**	0.2 ±0.0	71.6
	expert	0.2 ±0.1	2.7 ±0.9	1.6 ±0.5	**6.1 ±3.7**	0.1 ±0.0	85.8

uncertainty as in Yu et al. (2020b). We adopt an analogous approach to that studied in Lu et al. (2022); Sekar et al. (2020) and use the mean-disagreement of the ensemble. Thus, the reward at each step becomes:

$$\tilde{r}(s,a) = r(s,a) - \lambda \sum_{k=1}^{K} (\mu_\phi^k(s,a) - \mu^*(s,a))^2, \tag{1}$$

where λ is a penalty weight and $\mu^*(s,a) = \frac{1}{K}\sum_{k=1}^{K} \mu_\phi^k(s,a)$ is the mean over the dynamics ensemble. Instead of interleaving model-training steps and policy optimization steps, we simply perform one phase of each. We refer to this algorithm as **Offline DV2**.

Model-free. For DrQ-v2, we note that the base policy optimizer shares similarities with TD3 (Fujimoto et al., 2018), which has recently been applied effectively in offline settings from proprioceptive states by simply adding a regularizing behavioral-cloning term to the policy loss, resulting in the algorithm TD3+BC (Fujimoto & Gu, 2021). Concretely, the policy objective becomes:

$$\pi = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{env}}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right], \tag{2}$$

where λ is a normalization term, Q is the learned value function and π is the learned policy. We apply the same regularization to DrQ-v2, and call this algorithm: **DrQ+BC**.

Prior work. Since DrQ-v2 is an actor-critic algorithm, we may also use it to readily implement the **CQL** (Kumar et al., 2020) algorithm by adding the CQL regularizers to the Q-function update. We additionally compare against **LOMPO** (Rafailov et al., 2021), and behavioral cloning (**BC**, Bain & Sammut (1995); Bratko et al. (1995)), where we apply supervised learning to mimic the behavioral policy. Offline DV2 is closely related to LOMPO as both use an RSSM (Hafner et al., 2019; 2020a) as the fundamental model, however, Offline DV2 is based on the newer discrete RSSM with an uncertainty penalty more closely resembling the ensemble penalties in supervised learning (Lakshminarayanan et al., 2017) and uses KL balancing during training (Hafner et al., 2020b).

Table 2: We confirm that our simple baselines outperform LOMPO on the original walker-walk data provided by Rafailov et al. (2021). We report final performance mapped from $[0, 1000]$ to $[0, 100]$ averaged over six random seeds.[1] We show our baselines are more performant than LOMPO on their benchmark. CQL were numbers taken from Rafailov et al. (2021).

LOMPO Dataset	LOMPO	Offline DV2	DrQ+BC	CQL
medium-replay	61.3 ±9.1	**76.3 ±3.1**	31.1 ±3.7	14.7
medium-expert	69.0 ±24.1	72.3 ±20.1	**73.3 ±3.5**	45.1
expert	52.4 ±35.7	59.4 ±26.6	**90.8 ±2.2**	40.3

4.3 Comparative Evaluation

We now evaluate the five algorithms described in Section 4.2 on a total of fifteen datasets. To provide a fair evaluation, we provide full training curves for each algorithm in Figure 2 and summarize final performance with error in Table 1. Since no online data collection is required, we measure progress through training via an "offline epochs" metric which we define in Appendix C.

Table 1 shows a clear trend: Offline DV2 is the strongest on the random and mixed datasets, consisting of lower-quality but diverse data, DrQ+BC is the best on datasets with higher-quality but still widely-distributed data and pure BC outperforms on the high-quality narrowly-distributed expert data. We see from Table 1 and Figure 2 that DrQ+BC is extremely stable across seeds and training steps and has the highest overall performance. CQL is also a strong baseline, especially on expert data, but requires significant hyperparameter tuning per dataset, often has high variance across seeds, and is also far slower than DrQ+BC to train. Finally, no algorithm achieves strong performance on the challenging humanoid datasets, mirroring the online RL challenges (Hafner et al., 2020b; Yarats et al., 2021a), with only the supervised BC and, by extension, DrQ+BC showing marginal positive returns.

Perhaps surprisingly, Offline DV2 learns mid-level policies from random data on DMC environments. Furthermore, the random data are more challenging than their D4RL equivalents because there is no early termination, and thus mostly consists of uninformative failed states; this shows the strength of model-based methods in extracting signal from large quantities of suboptimal data. On the other hand, Offline DV2 is considerably weaker on the expert datasets that have narrow data distributions. For these environments, we find the uncertainty penalty is uninformative, as discussed in Appendix D.4.1.

Taking all these findings into consideration leads us to our first open problem, which we believe continued research using our benchmark can help to answer:

Challenge 1: Can a single algorithm outperform both the model-free and model-based baselines, and produce strong performance across *all* offline datasets?

4.3.1 Comparison to the LOMPO Benchmark

For a fair comparison to LOMPO, we also benchmark on the data used in Rafailov et al. (2021) on the DMC Walker-Walk task. In the LOMPO benchmark, the datasets are limited to three types: {medium-replay, medium-expert and expert}. We provide final scores in Table 2. While LOMPO struggles on V-D4RL, it performs reasonably on its own benchmark. However, LOMPO is still outperformed by Offline DV2 on all datasets, whereas DrQ+BC is the best on two datasets.

We may explain the relative strength of LOMPO on this benchmark by noting that the medium-expert dataset used by Rafailov et al. (2021) is described as consisting of the second half of the replay buffer after the agent reaches medium-level performance, thus containing far more diverse data than a bimodal D4RL-style concatenation of two datasets. Furthermore, the expert data is far more widely distributed than that of a standard SAC expert, as we confirm in the statistics in Table 6 of Appendix A.

[1] It is unclear how the original scores in Rafailov et al. (2021) were normalized.

Figure 3: Both DrQ+BC and Offline DV2 readily support training datasets with different distractions (mixture of original and shifted train). Offline DV2 additionally shows the ability to generalize to unseen distractions (shifted test) whereas DrQ+BC is more brittle. Return is normalized against unshifted performance without distractions from Table 1 and averaged over six random seeds. Unnormalized returns are provided in Tables 11 and 12 in Appendix D.1.

5 Desiderata for Offline RL from Visual Observations

A key ingredient in recent advances in deep learning is the use of large and diverse datasets, which are particularly prevalent in vision, to train models. Such datasets enable the learning of *general* features that can be successfully transferred to downstream tasks, even when the original task bears little immediate similarity with the transferred task (Xie & Richmond, 2019; Peters et al., 2019). This is a clear rationale for adopting visual observations in offline RL; by leveraging large quantities of diverse high-dimensional inputs, we should be able to learn generalizable features and agents for control. However, combining rich visual datasets with RL presents its own unique challenges. In this section, we present important desiderata that highlight this, and conclude each with an open problem that requires further research.[2]

5.1 Desideratum: Robustness to Visual Distractions

One desideratum for offline RL from visual observations is the ability to learn a policy from data collected under multiple different settings. For example, quadrupedal robots deployed at different times of day (e.g., morning and night) will likely gather data having significantly different visual properties, such as brightness and range of visibility. Although the robot may produce similar behaviors in both deployments, superficial differences in visual outputs present a host of opportunities for the agent to learn spurious correlations that prevent generalization (Song et al., 2020; Raileanu & Fergus, 2021).

A key opportunity that arises is the potential to *disentangle* sources of distractions through training on multiple settings, facilitating the learning of general features. By contrast, proprioceptive observations do not generally have distractions, as they are typically low-dimensional and engineered for the task of interest (Lesort et al., 2018). This also limits their ability to transfer, as it is unclear how to incorporate features learned under one set of agent geometries to another.

To test a simplified version of this challenge, we train our baseline agents using newly created datasets featuring varying visual augmentations from the Distracting Control Suite (Stone et al., 2021). This suite provides three levels of distractions (i.e., low, moderate, high), and each distraction represents a shift in the data distribution. We subsequently refer to the level of distraction as the "shift severity" (Schneider et al., 2020). The offline datasets are then constructed as mixtures of samples from the original environment

[2]As CQL is quite sensitive to hyperparameters per environment, in the following sections we use the more robust Offline DV2 and DrQ+BC algorithms.

without distractions and samples from an environment with a *fixed distraction* level. The learned policies are then evaluated on test environments featuring unseen distractions of the same shift severity.

To create these datasets, we note that the visual distractions from the Distracting Control Suite do not manifest themselves in the proprioceptive states, so we can naturally generate the shifted visual observations by simply rendering the distractions on top of the existing visual observations. Thus, we can simply use the same saved checkpoints as in Section 4.1.1 for the standard datasets.

We compare the baseline algorithms, Offline DV2 and DrQ+BC on datasets that they excel on in Section 4 and Section 5.3: walker-walk random with 100,000 datapoints and cheetah-run medium-expert with 1 million respectively. We visualize returns normalized by this unshifted performance in Figure 3 to show the generalization of each algorithm. We further present full tabular results for Offline DV2 and DrQ+BC in Table 11 and Table 12 respectively in Appendix D.1. The highlighted base unshifted results in both tables are the same as in Table 1 for Offline DV2 and Table 4 for DrQ+BC.

Offline DV2 is able to accommodate datasets with mixed distractions and generalizes reasonably well to unseen test distractions, especially when trained with 'low' and 'moderate' levels of shift severity. Similarly, DrQ+BC is robust to multiple training distractions, with little to no degradation in performance on mixed datasets. However, in contrast, the policy learned is brittle with respect to unseen distractions, and performs significantly worse on the test environments.

Overall, both Offline DV2 and DrQ+BC represent strong baselines for mixed datasets. Interestingly, Offline DV2 demonstrates strong generalization to unseen distractions. This can be explained by generalization that occurs in the trained world-model, which uses a self-supervised loss; we discuss reasons behind this in Appendix D.5. This could be improved even further with recent reconstruction-free variants of DreamerV2 (Okada & Taniguchi, 2021; Nguyen et al., 2021) which have shown robustness to distractions. On the other hand, we observe DrQ+BC generalizes poorly to unseen distractions, presenting a direction for future work using our datasets to learn robust *model-free* agents. This directly leads us to our next open problem:

Challenge 2: How can we improve the generalization performance of offline model-free methods to unseen visual distractions?

5.2 Desideratum: Generalization Across Environment Dynamics

Another desideratum of offline RL from visual observations is learning policies that can generalize across multiple dynamics. This challenge manifests in three clear ways. Firstly, we will likely collect data from multiple agents that each have different dynamics, and must therefore learn a policy that can perform well when deployed on any robot that gathered the data (i.e., train time robustness). Secondly, we may be provided with asymmetric data, featuring scarce coverage of particular dynamics, and therefore require the

Table 3: Evaluation on the DMC-Multitask benchmark using *random* data for Offline DV2 and *medexp* data for DrQ+BC and BC. Normalized performance from [0, 1000] to [0, 100] over six random seeds. Our algorithms learn multitask policies from visual observations, with a slight generalization gap for extrapolation tasks. Different dataset types are used for each algorithm to reflect realistic use cases.

Algorithm	**Environment**		**Eval. Return**		
			Train Tasks	**Test Interp.**	**Test Extrap.**
DrQ+BC	walker	medexp	90.8	91.4	65.1
	cheetah		71.6	65.1	43.2
BC	walker		61.2	61.4	47.2
	cheetah		69.7	61.3	39.6
Offline DV2	walker	random	24.4	25.3	24.9
	cheetah		31.6	31.1	31.1

Table 4: The reinforcement learning algorithms readily scale to higher dataset sizes, compared to supervised behavioral cloning, with a particular benefit to the *medexp* and *expert* datasets for DrQ+BC and the *random* and *medium* datasets for Offline DV2. Results are averaged over six random seeds, with one standard deviation given as error. The evaluated return is mapped from $[0, 1000]$ to $[0, 100]$, and the fixed-size *mixed* dataset is excluded.

Environment		**Offline DV2**		**DrQ+BC**		**BC**	
		100K	**500K**	**100K**	**500K**	**100K**	**500K**
walker	random	28.7 ±13.0	49.9 ±1.6	5.5 ±0.9	3.5 ±0.6	2.0 ±0.2	2.1 ±0.3
	medium	34.1 ±19.7	61.3 ±10.9	46.8 ±2.3	51.0 ±1.1	40.9 ±3.1	40.9 ±3.0
	medexp	43.9 ±34.4	38.9 ±28.1	86.4 ±5.6	94.1 ±2.0	47.7 ±3.9	48.8 ±5.3
	expert	4.8 ±0.6	7.1 ±5.3	68.4 ±7.5	94.2 ±2.3	91.5 ±3.9	95.1 ±2.5
cheetah	random	31.7 ±2.7	40.8 ±4.2	5.8 ±0.6	10.6 ±0.7	0.0 ±0.0	0.0 ±0.0
	medium	17.2 ±3.5	39.2 ±14.4	53.0 ±3.0	57.3 ±1.2	51.6 ±1.4	52.9 ±1.3
	medexp	10.4 ±3.5	9.7 ±5.0	50.6 ±8.2	79.1 ±5.6	57.5 ±6.3	69.6 ±10.6
	expert	10.9 ±3.2	11.3 ±4.7	34.5 ±8.3	75.3 ±7.5	67.4 ±6.8	87.8 ±1.9
Average Overall		22.7 ±10.1	32.3 ±9.3	43.9 ±4.6	58.1 ±2.6	44.8 ±3.2	49.7 ±3.1
Percentage Gain		+42.1%		+32.5%		+10.8%	

ability to leverage data from more abundant sources (i.e., transferability). Thirdly, we may be presented with *unseen* dynamics at deployment time, and must therefore learn a policy that is robust to these changes (i.e., test time robustness).

A key opportunity that arises in visual observations is the improved richness of the underlying dataset compared to proprioceptive data. For instance, *some dynamics changes may be visually obvious* (e.g., changed limb sizes, broken actuators), whereas in the proprioceptive setting, such information may not be available. Without this information, we must turn to meta-RL (Rakelly et al., 2019; Zintgraf et al., 2021) or HiP-MDP (Killian et al., 2017; Zhang et al., 2021) approaches that try to infer the missing information from gathered trajectories, adding complexity to the RL process. In contrast, this information can be contained explicitly in visual observations and should allow adaption to a range of downstream tasks without complex inference methods.

To test this hypothesis, we consider two settings from the MTEnv benchmark (Sodhani et al., 2021) which adapts DMC: cheetah-run with modified torso length and walker-walk with modified leg length. We follow an analogous approach to Zhang et al. (2021) where we consider eight different settings {A - H} ordered in terms of limb length, and construct new offline datasets using {B, C, F, G} in equal proportions as our training data. The settings {A, H} are considered the *extrapolation* generalization environments and {D, E} *interpolation* generalization. Since these environments are different from the original, we retrain SAC policies on the proprioceptive states from the modified tasks and define medium and expert policies in the same way as described in Section 4.1.1.

As before, to provide a comparison on realistic use cases of each algorithm, we evaluate Offline DV2 on random datasets of size 100,000 and DrQ+BC on medium-expert datasets containing one million samples and show the results in Table 3. We see that DrQ+BC learns policies that are suitable for transfer across multiple tasks in both walker and cheetah, and maintains that performance on the interpolation test environments. For the extrapolation environments, we see an average drop of around 30%. While this may represent adequate performance, especially compared to a medium policy, it is a striking drop when compared to performance on in-distribution dynamics. This suggests there is a dynamics generalization gap that remains for model-free methods when extrapolating, and represents clear opportunities for further research.

Offline DV2 displays similar trends (results on medexp datasets are in Appendix D.2). On the random data, Offline DV2 learns a similar quality policy to that on the base environment, but experiences no deterioration in performance on the test environments in walker or cheetah. Thus, we demonstrate the sufficiency of

both Offline DV2 and DrQ+BC as baselines in multitask offline RL *without any modification*; model-based approaches further admit opportunities for zero-shot generalization (Ball et al., 2021).

We now contrast our work to that of Zhang et al. (2021), where a multitask policy was trained using a total of 3.2 million time steps of online data collection. Whilst it is hard to compare offline and online results, our DrQ+BC algorithm uses less data, with 1 million total time steps, and obtains similar extrapolation return on the walker environments. This supports a similar conclusion reached by Kurin et al. (2022) who show that our approach, simply minimizing the sum of the task losses, is drastically underestimated in the literature. As noted before, we suffer a comparatively larger drop in performance, lending further evidence that closing this generalization gap should be prioritized.

In conclusion, we believe there are many further avenues for future research using these benchmarks; an immediate open problem we have identified is as follows:

Challenge 3: How can we improve generalization to new dynamics that are not contained in the offline dataset?

5.3 Desideratum: Improved Performance with Scale

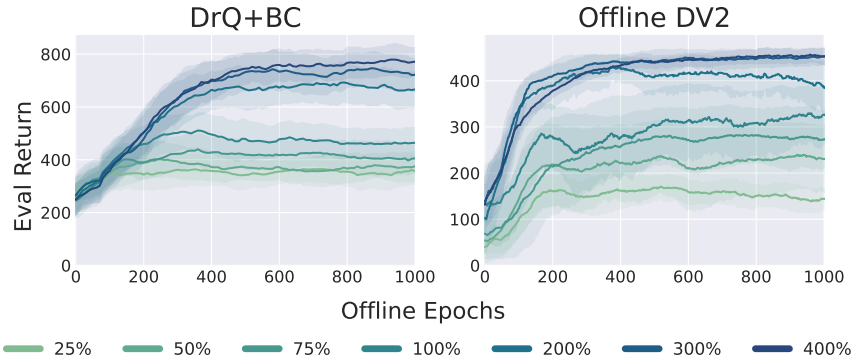


Figure 4: Sensitivity analysis on dataset size for both Offline DV2 (walker-walk random) and DrQ+BC (cheetah-run medexp). Both methods scale with more data, but receive diminishing returns past 2× the original size. Performance averaged over four random seeds.

Learning from large datasets presents huge opportunities for learning general agents for control. To make use of them, we need to understand how our baselines scale with dataset size. We analyze our base choice of 100,000 observations for V-D4RL in Figure 4, where we vary the size of the walker-walk random dataset for Offline DV2 and the cheetah-run medexp dataset for DrQ+BC in the range of $\{0.25\times, \ldots, 4\times\}$ the size of the original dataset. We observe a monotonic increase in the performance of both Offline DV2 and DrQ+BC with increasing dataset size but hit diminishing returns past 2× the original size. We note, perhaps surprisingly, that Offline DV2 can reach ≈500 total return from random data that average 10× less.

For the walker and cheetah datasets with fixed distributions—random, medium, medium-expert and expert—Table 4 shows an average increase of 42.1% for Offline DV2 and 32.5% for DrQ+BC compared to 10.8% for BC when we scale the dataset size to 500,000, showing that the reinforcement learning algorithms make far better use of increased data than supervised behavioral cloning. However, a crucial difference between Offline DV2 and DrQ+BC is that DrQ+BC handles larger offline datasets far more readily. DrQ+BC policy training for the same number of epochs on 500,000 and 100,000 observations takes 8 and 1.6 hours respectively on a V100 GPU. This is significantly quicker than Offline DV2, which takes 10 hours to train on 100,000 observations; we discuss this further in Appendix D.4.1. This significant computational discrepancy leads to a clear open problem:

Challenge 4: How can we scale model-based methods to larger datasets?

6 Conclusion and Limitations

In this paper, we took the first steps towards establishing fully open-sourced benchmarking tasks featuring a broad range of behavioral policies and competitive baselines for offline reinforcement learning from visual observations. Until now, work in this space has been nascent, with ad-hoc analyses leading to unclear comparisons. To address the lack of meaningful evaluations and comparative analyses in this space, we provided a set of straightforward and standardized benchmarking tasks that follow popular low-dimensional equivalent experiment setups and presented competitive model-based and model-free baselines. We analyzed key factors that help explain the performance of these approaches, while also demonstrating their ability to generalize in more challenging settings that are unique to visual observations.

With a particular focus on the DeepMind Control Suite, which features a wide array of tasks of varying difficulties, we hope that V-D4RL will be useful to practitioners and researchers alike and that it will provide a springboard for developing offline reinforcement learning methods for real-world continuous-control problems and spark further progress in this space. Complementing and extending V-D4RL to more domains that feature complex manipulation tasks and physical robotics systems presents an exciting direction for future research.

Acknowledgements

Cong Lu and Tim G. J. Rudner are funded by the Engineering and Physical Sciences Research Council (EPSRC). Philip J. Ball is funded through the Willowgrove Studentship. Tim G. J. Rudner is also funded by a Qualcomm Innovation Fellowship. We are grateful to Rafael Rafailov for sharing the data used with LOMPO and Edoardo Cetin for his NumPy Array replay buffer implementation. We thank Samuel Daulton, Clare Lyle, and Muhammad Faaiz Taufiq for detailed feedback on an early draft of this paper. The authors would also like to thank the anonymous TMLR reviewers for constructive feedback that helped improve the paper.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 619–629. PMLR, 2021.

Ivan Bratko, Tanja Urbančič, and Claude Sammut. Behavioural cloning: Phenomena, results and problems. *IFAC Proceedings Volumes*, 28(21):143–149, September 1995. ISSN 1474-6670. doi: 10.1016/s1474-6670(17)46716-4.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loïc Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *CoRR*, abs/1804.03599, 2018.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.076.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan C Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1518–1528. PMLR, 18–24 Jul 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 6(18):503–556, 2005.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *5th Annual Conference on Robot Learning*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning, 2021.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062, 2019.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando deFreitas. Rl unplugged: Benchmarks for offline reinforcement learning, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, pp. 2555–2565, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020a.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020b.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825): 357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

Arnav Kumar Jain, Shiva Kanth Sujit, Shruti Joshi, Vincent Michalski, Danijar Hafner, and Samira Ebrahimi Kahou. Learning robust dynamics through variational sparse gating. In *Deep RL Workshop NeurIPS 2021*, 2021.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 29–31 Oct 2018.

Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823. Curran Associates, Inc., 2020.

Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *Deep RL Workshop NeurIPS 2021*, 2021.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning, 2022.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19884–19895. Curran Associates, Inc., 2020.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Francois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, December 2018. ISSN 0893-6080. doi: 10.1016/j.neunet.2018.07.006.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

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 Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Cong Lu, Philip J. Ball, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, November 2016. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364916679498.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4402–4412. PMLR, 09–15 Jun 2019.

Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8130–8139. PMLR, 18–24 Jul 2021.

D.A. Nix and A.S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pp. 55–60 vol.1. IEEE, 1994. doi: 10.1109/icnn.1994.374138.

Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4209–4215, 2021. doi: 10.1109/ICRA48506.2021.9560734.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pp. 7–14, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/w19-4302.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pp. 1154–1168. PMLR, 2021.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8787–8798. PMLR, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pp. 5331–5340. PMLR, 2019.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *arXiv preprint arXiv:2006.16971*, 2020.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://arxiv.org/abs/2012.09812.

Shagun Sodhani, Ludovic Denoyer, Pierre-Alexandre Kamienny, and Olivier Delalleau. MTEnv - environment interface for mulit-task reinforcement learning. Github, 2021.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. Springer US, second edition, 1992. ISBN 9781461366089, 9781461536185. doi: 10.1007/978-1-4615-3618-5.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm_control: Software and tasks for continuous control, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019.

Yiting Xie and David Richmond. Pre-training on grayscale ImageNet improves medical image classification. In Laura Leal-Taixé and Stefan Roth (eds.), *Computer Vision – ECCV 2018 Workshops*, pp. 476–484, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11024-6.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021b.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021c.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020a. doi: 10.1109/cvpr42600.2020.00271.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142. Curran Associates, Inc., 2020b.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization, 2021.

Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Learning robust state abstractions for hidden-parameter block MDPs. In *International Conference on Learning Representations*, 2021.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: Variational Bayes-adaptive deep RL via meta-learning. *Journal of Machine Learning Research (JMLR)*, 22(289):1–39, 2021.

Limitations and Future Work

As we concluded in the paper, we hope that our work helps facilitate progress in the burgeoning field of offline reinforcement learning from visual observations. Of particular interest to us are the open questions and challenges we highlighted in the paper, namely:

1. Can a single algorithm outperform both the model-free and model-based baselines and produce strong performance across *all* offline datasets?
2. How can we improve the generalization performance of our algorithms to unseen visual distractions and dynamics?
3. How can we scale model-based methods to larger datasets?

Indeed, we are particularly encouraged that recent work (Ball et al., 2023; Islam et al., 2022; Tarasov et al., 2023; Zang et al., 2022) has begun to tackle these questions as well as use our benchmark as a springboard for new ideas.

One limitation of our work is that our benchmarks and analyses are focused on simulated DMControl Suite tasks. Exciting future work could include testing our approaches on more complex manipulation tasks and physical robotics systems. We have good reason to believe that this is possible given the right dataset, as online variants of the algorithms we use (Wu et al., 2022) have been successful in controlling real-world robots. Another harder target for our algorithms could be the MineRL datasets (Guss et al., 2019) which include a vast set of human demonstrations for various tasks in the Minecraft game world.

Finally, we note that expanding capabilities in learning from visual data also takes us closer to the under-explored field of learning from multi-modal data. Hansen et al. (2022) show that combining image and proprioceptive observations can enable rapid learning of complex visuomotor control tasks. Future work in offline multi-modal learning could be useful for settings like autonomous driving for which a lot of pre-collected data exists with complex observation spaces that could include: video, LIDAR and RADAR sweeps, and map data (Caesar et al., 2020).

8

Conclusion

Contents

In this thesis, we developed a series of methods that aimed to improve the data efficiency and robustness of agents trained via reinforcement learning. While reinforcement learning has seen stunning successes in sequential decision-making problems, it has remained difficult to deploy in general due to its notorious sample inefficiency and brittleness in real-world scenarios.

To address these challenges, the thesis was divided into two parts, focusing on complementary approaches to augmenting the data that agents are normally trained on. Part I delved into the development of synthetic data and environments as an additional source of useful data. In Chapter 3, we began by demonstrating that existing approaches are unable to adapt to changing dynamics at test time, particularly if they were trained on data from only a single environment. We introduced

Augmented World Models, an approach that enabled agents to generalize zero-shot to novel dynamics by augmenting learned dynamics models with transformations that sought to capture potential test time changes. Furthermore, we believe that we are the first to introduce this formulation, which enables agents to **meta-train over a rich distribution of imagined environments**. We are particularly excited about extensions of this approach to capture broader families of augmentations, including semantic augmentations enabled by recent advances in generative video modeling.

Chapter 4 proposed a novel approach to generating synthetic data for reinforcement learning, *Synthetic Experience Replay*, which leverages generative modeling to directly upsample agent training data. We demonstrated the ability of modern diffusion models to accurately generate synthetic experiences across six distinct proprioceptive and pixel-based algorithms, **with no algorithmic modification**. Particular highlights of our approach in offline reinforcement learning include the ability to train from extremely small datasets, scaling up policy and value networks, and high levels of data compression. In online reinforcement learning, the additional data allows agents to use much higher update-to-data ratios than before, leading to increased sample efficiency. We have only scratched the surface of what is possible in this space, and believe further work in guided diffusion or fine-tuning from pretrained diffusion models could unlock entirely new training strategies.

Part II centered around learning from pre-collected offline data. In Chapter 5, we first considered the hybrid setting where we seek to accelerate online reinforcement learning given a dataset of expert demonstrations. We identified a previously unrecognized pathology in KL-regularized RL from expert demonstrations due to the **poor uncertainty quantification of parametric behavioral policies** and showed that this pathology can significantly impede and even entirely prevent online learning. To remedy the pathology, we proposed the use of non-parametric behavioral reference policies in the algorithm *Non-Parametric Prior Actor–Critic*, which led to a new state-of-the-art in a variety of challenging continuous control

tasks. Moreover, we hope this work draws attention to the use of better model classes in deep reinforcement learning, where neural networks are the predominant choice.

In Chapter 6, we rigorously evaluated the impact of various key design choices in offline model-based reinforcement learning, and compared for the first time various uncertainty heuristics that have been proposed in the literature to address distribution shift. By proposing novel evaluation protocols, we discovered key insights into the role of uncertainty and key hyperparameter choices that we hope will benefit the wider community. Furthermore, we put our insights into practice and proposed to automatically select configurations in this design space with a powerful Bayesian Optimization agent. By doing so, we showed that we can produce **superior configurations with vastly different key hyperparameters** to existing algorithms and significant performance improvements in almost all benchmarks.

Finally, in Chapter 7, we established the first open-sourced comprehensive benchmarking suite and competitive baselines in offline reinforcement learning from visual observations, *Vision Datasets for Deep Data-Driven RL*. Our benchmark aimed to replicate popular low-dimensional equivalents. Together with two novel model-based and model-free offline baselines that we construct from standard online algorithms, we establish a set of meaningful evaluations and comparative analyses in this space. Furthermore, we analyzed key factors that help explain the performance of our algorithms, while also demonstrating their ability to generalize in more challenging settings that are unique to visual observations. We hope that this work will lead to further progress in this burgeoning field and **provide a springboard for developing offline reinforcement learning methods from visual observations**.

8.1 A Roadmap to Efficient and Robust Agents

As we look to the future, we connect the fields of synthetic and offline data to exciting recent developments in supervised learning and envision how they could contribute to the development of more generally capable agents.

8.1.1 Synthetic Experience and Large Generative Foundation Models

It has long been posited that humans owe their ability to adapt to unseen or complex situations by leveraging internal predictive models of the world (Craik, 1967). LeCun (2022) argues that such models allow us to predict the consequences of our actions without the need for large numbers of dangerous trials in the real world. Furthermore, with these models, we can reason, plan, and imagine new solutions to novel problems without any interaction with the environment. The advent of large generative foundation models (Ho et al., 2022; Hu et al., 2023) has begun to make this a reality for real-world situations.

Particularly notable is the video foundation model GAIA-1 (Hu et al., 2023) which can generate realistic driving scenes from a video prompt, and also be further conditioned on intended sequences of actions or text prompts like "we are behind a bus". As we alluded to at the end of Chapter 3, such models could enable *semantic augmentations* to existing training environments, even adversarially targeting scenarios where a particular learned agent was weak. One could liken this procedure to the human process of mentally rehearsing a new plan before execution. Even without additional training, synthetic rollouts could already give hints about the failure modes of existing agents (Igl et al., 2022).

One could also reap the benefit of foundation models without considering planning. Another recent trend in supervised learning is the ability to fine-tune (Houlsby et al., 2019; Hu et al., 2022; Li and Liang, 2021; Zhang et al., 2023b) or adapt pre-trained foundation models towards specific tasks. This is often orders of magnitude cheaper than training from scratch and can allow a model to retain broad knowledge from

the pre-training stage. While in Chapter 4, we investigated training a generative model from scratch to upsample training data, fine-tuning a text-image foundation model like Stable Diffusion (Rombach et al., 2022) could drastically reduce the requirements for real environment data.

Furthermore, using such a generative model could also allow us to *generalize across concepts learned during the pre-training stage*. Recent work (Chen et al., 2023; Yu et al., 2023) has already shown progress towards enabling robotic arms to be able to pick any object in any unseen environment (Yenamandra et al., 2023) by generating novel combinations of objects with various prompting strategies. This will only improve with more advanced generative models that allow faithful finer-grained control from text prompts (Betker et al., 2023).

8.1.2 Leveraging Internet-Scale Data

The field of offline reinforcement learning naturally connects to recent drives in leveraging internet-scale data (OpenAI, 2023; Rombach et al., 2022; Touvron et al., 2023) for large generative models. In particular, one of the north stars of offline reinforcement learning would be to distill vast amounts of diverse pre-existing multitask data into a single generalist agent (Reed et al., 2022) that could operate across different modalities and embodiments. Indeed, the recent Robotics Transformer series (Brohan et al., 2023a,b; Vuong et al., 2023) has shown the potential of such a paradigm, combining powerful pre-trained vision-language models with diverse datasets sourced from dozens of research labs across the world.

For example, the RT-2 (Brohan et al., 2023a) family of models cast actions as text tokens and thus is able to fine-tune any pre-trained vision-language model to control a robot in a closed-loop manner. While the tasks we consider in Chapters 5, 6, and 7 of the thesis are not language-conditioned, we could similarly consider pre-trained representations to annotate the observations in our offline data. This could be particularly effective for the visual observations in Chapter 7 and provide a solution for the challenging Humanoid datasets.

Furthermore, we are also excited by recent work using large language models to drive exploration in open-ended environments. For example, Hu and Clune (2023); Wang et al. (2023); Zhang et al. (2023a) propose methods to use large language models to generate new goals or high-level plans that an agent should follow. However, realizing these plans still requires some form of goal-directed reward signal or environment scripting language, and as such these methods are not fully general.

We believe that recent work in deriving reward functions from large language models (Klissarov et al., 2023; Kwon et al., 2023) based on adjustable text prompts could remove these requirements. We envision a synthesis of the above ideas where, in an outer loop, a large language or multimodal model can both propose a relevant goal along with a reward function to evaluate its completion. In an inner loop, a reinforcement learning agent, possibly a small head on top of a pre-trained foundation model, could then learn to optimize that reward function. This could lead to Voyager-style open-ended discovery of new skills but in any environment or action space.

8.2 Outlook for the Future

Future advances in these two directions could lead us to human-level capabilities, where agents could combine vast amounts of general knowledge of the world with robust planning and notions of curiosity to drive life-long learning. We view this process as fundamentally tied to modeling the world and making use of large amounts of prior data, which is the overall theme of this thesis.

We are excited and optimistic about what advances in reinforcement learning might bring, especially in the age of large generative multimodal models. We believe that our work contributes new ways to use data, synthetic or offline, to train agents and hope that this may spark further ideas in the field. However, as agents become more generally capable and begin to be deployed in the real world, we must also

consider the societal and ethical implications of doing so. A few of these risks are outlined here.

As with all technological advancements that increase the capabilities of humans, machine learning has the potential to amplify both beneficial and detrimental applications. Furthermore, unequal distribution of any technological advancement tends to cause severe losses in jobs (Frey and Osborne, 2017) and further deepens economic inequality (Atkinson, 2015; Links, 2006). More specific to modern machine learning, is the fact that while capable, current trained agents are often uninterpretable (Carvalho et al., 2019) and difficult to diagnose. This exacerbates the risk that an agent trained only to perform a particular task may produce unexpected consequences (Zhuang and Hadfield-Menell, 2020). Current systems are also extremely prone to amplify systematic biases in data, leading to issues of fairness (Barocas et al., 2017; Mehrabi et al., 2021).

Finally, far into the future, we face the prospect of developing agents more generally intelligent than us. It is unclear what this means for humanity, with some believing that controlled intelligence of this form could enable us to resolve the remaining open questions in science and alleviate resource scarcity. Others believe that such intelligence would be fundamentally uncontrollable, and has the potential to lead us on the path to extinction (Hendrycks and Mazeika, 2022). As we are in uncharted territory, we must be proactive in establishing robust safety measures and effective governance mechanisms to ensure that our field plays a positive role.

Bibliography

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=nIAxjsniDzg`.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009.

Anthony B Atkinson. *Inequality: What can be done?* Harvard University Press, 2015.

Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J. Fleet. Synthetic data from diffusion models improves imagenet classification, 2023.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In Armand Prieditis and Stuart Russell, editors, *Machine*

Learning Proceedings 1995, pages 30–37. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50013-X. URL https://www.sciencedirect.com/science/article/pii/B978155860377650013X.

Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 591–601. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/ball20a.html.

Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 619–629. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ball21a.html.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Albert Bandura. Observational learning. *The international encyclopedia of communication*, 2008.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2017, 2017.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M Zhang. Human-timescale adaptation in an open-ended task space. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1887–1935. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/bauer23a.html`.

Normand J. Beaudry and Renato Renner. An intuitive proof of the data processing inequality, 2012.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. 2023.

Herman J. Bierens. *The Nadaraya–Watson kernel regression function estimator*, page 212–247. Cambridge University Press, 1994. doi: 10.1017/CBO9780511599279.011.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

David Brandfonbrener, Remi Tachet des Combes, and Romain Laroche. Incorporating explicit uncertainty estimates into deep offline reinforcement learning, 2022.

Ivan Bratko, Tanja Urbancic, and Claude Sammut. Behavioural cloning: phenomena, results and problems. *IFAC Proceedings Volumes*, 28(21):143–149, 1995.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted

Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023b.

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, 2020.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

Alan Chan, Hugo Silva, Sungsu Lim, Tadashi Kozuno, A. Rupam Mahmood, and Martha White. Greedification operators for policy optimization: Investigating forward and reverse kl divergences. *J. Mach. Learn. Res.*, 23(1), jan 2022. ISSN 1532-4435.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021a.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021b. URL `https://openreview.net/forum?id=AY8zfZm0tDd`.

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in AlphaGo. *CoRR*, abs/1812.06855, 2018.

Yutian Chen, Liyuan Xu, Caglar Gulcehre, Tom Le Paine, Arthur Gretton, Nando de Freitas, and Arnaud Doucet. On instrumental variable regression for deep offline policy evaluation, 2021c.

Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv:2302.06671*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence, 2020.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, 2006. URL `https://api.semanticscholar.org/CorpusID:16724115`.

Kenneth James Williams Craik. *The nature of explanation*, volume 445. CUP Archive, 1967.

Andreas Damianou and Neil D. Lawrence. Deep Gaussian processes. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR. URL https://proceedings.mlr.press/v31/damianou13a.html.

C. Darwin, P. Ekman, and P. Prodger. *The Expression of the Emotions in Man and Animals*. Oxford University Press, 1998. ISBN 9780195158069. URL https://books.google.co.uk/books?id=TFRtLZSHMcYC.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414–419, 02 2022. doi: 10.1038/s41586-021-04301-9.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019. URL https://openreview.net/forum?id=S1xtR52NjN.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 6(18): 503–556, 2005.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge, 2022.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 10 2022. doi: 10.1038/s41586-022-05172-4.

Angelos Filos, Clare Lyle, Yarin Gal, Sergey Levine, Natasha Jaques, and Gregory Farquhar. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In *International Conference on Machine Learning*, pages 3305–3317. PMLR, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *5th Annual Conference on Robot Learning*, 2021.

Carl Benedikt Frey and Michael A Osborne. The future of employment: How susceptible are jobs to computerisation? *Technological forecasting and social change*, 114:254–280, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, ziyu wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Thomas Paine. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=kWSeGEeHvF8`.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/fujimoto18a.html`.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/fujimoto19a.html`.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Vinicius G Goecks, Gregory M Gremillion, Vernon J Lawhern, John Valasek, and Nicholas R Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. *arXiv preprint arXiv:1910.04281*, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019. URL `http://minerl.io`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, pages 2555–2565, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020a.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020b.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18100–18115. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/72dad95a24fae750f8ab1cb3dab5e58d-Paper-Conference.pdf`.

Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint*, 2022.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Dan Hendrycks and Mantas Mazeika. X-risk analysis for ai research. *arXiv preprint arXiv:2206.05862*, 2022.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian

Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations, 2017. URL https://arxiv.org/abs/1704.03732.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Shengran Hu and Jeff Clune. Thought Cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 2023.

Marcus Hutter. A theory of universal artificial intelligence based on algorithmic complexity, 2000.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005. ISBN 3-540-22139-5. doi: 10.1007/b138233. URL http://www.hutter1.net/ai/uaibook.htm.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Qun8fv4qSby.

Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. *arXiv*, 2022.

Faical Isbaine, Marie Demolliens, Abdelouahed Belmalih, Andrea Brovelli, and Driss Boussaoud. Learning by observation in the macaque monkey under high experimental constraints. *Behavioural brain research*, 289:141–148, 2015.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm van Seijen, Remi Tachet des Combes, et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Matthew Jackson, Michael Matthews, Cong Lu, Jakob Foerster, and Shimon Whiteson. Policy-guided diffusion. In *6th Robot Learning Workshop NeurIPS 2023: Pretraining, Fine-Tuning, and Generalization with Large Scale Models*, 2023. URL https://openreview.net/forum?id=WXoTuoIxiC.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan,

Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13 (4):455–492, 1998.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=k7FuTOWMOc7`.

Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/f7efa4f864ae9b88d43527f4b14f750f-Paper.pdf`.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey, 2021.

Matthieu Kirchmeyer, Yuan Yin, Jeremie Dona, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11283–11301. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/kirchmeyer22a.html.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback, 2023.

W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis)design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103829. URL https://www.sciencedirect.com/science/article/pii/S0004370222001692.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, Anurag Daram, Stefano Fusi, Peter Helfer, Leslie Kay, Nicholas Ketz, Zsolt Kira, Soheil Kolouri, Jeff Krichmar, Sam Kriegman, and

Hava Siegelmann. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4:196–210, 03 2022. doi: 10.1038/s42256-022-00452-0.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Yann LeCun. A path towards autonomous machine intelligence. *Open Review*, 62, 2022.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 5757–5766. PMLR, 2020.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

Felix Leibfried, Vincent Dutordoir, ST John, and Nicolas Durrande. A tutorial on sparse gaussian processes and variational inference, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL `https://arxiv.org/abs/2005.01643`.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Fed Links. Economic inequality in the united states. *FRBSF Economic Letter*, 2006:33–34, 2006.

Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J. Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022a. URL `https://openreview.net/forum?id=zz9hXVhf40`.

Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations, 2022b. URL `https://arxiv.org/abs/2206.04779`.

Cong Lu, Philip J. Ball, and Jack Parker-Holder. Synthetic experience replay. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023. URL `https://openreview.net/forum?id=0a9p3Ty2k_`.

Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4170, 2021. doi: 10.1109/ICRA48506.2021.9561805.

Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, November 2016. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364916679498.

Daniel Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618:257–263, 06 2023. doi: 10.1038/s41586-023-06004-9.

Alex Martin. The representation of object concepts in the brain. *Annu. Rev. Psychol.*, 58:25–45, 2007.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

JM Mendel and RW McLaren. 8 reinforcement-learning control and pattern recognition systems. In *Mathematics in science and engineering*, volume 66, pages 287–318. Elsevier, 1970.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015. doi: 10.1038/nature14236.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*,

page 6292–6299. IEEE Press, 2018. doi: 10.1109/ICRA.2018.8463162. URL https://doi.org/10.1109/ICRA.2018.8463162.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021.

OpenAI. Gpt-4 technical report, 2023.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JYtwGwIL7ye.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings*

of Machine Learning Research, pages 17473–17498. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Pv1GPQzRrC8.

Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 1154–1168. PMLR, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR, 09–15 Jun 2019a. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019b.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=1ikK0kHjvj. Featured Certification, Outstanding Certification.

Marc Rigter, Jun Yamada, and Ingmar Posner. World models via policy-guided trajectory diffusion, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Ludan Ruan, Yiyang Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *CVPR*, 2023.

Tim G. J. Rudner, Cong Lu, Michael Osborne, Yarin Gal, and Yee Whye Teh. On pathologies in KL-regularized reinforcement learning from expert demonstrations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=sS8rRmgAatA.

Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractable function-space variational inference in bayesian neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=0Qs0pLKGGpS.

M. Sadler, N. Regan, and G. Kasparov. *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, 2019. ISBN 9789056918187. URL `https://books.google.co.uk/books?id=KhhivwEACAAJ`.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration, 2018.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes, 2017.

Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.05952`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/schulman15.html`.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b.

Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985. ISSN 0022-247X. doi: https://doi.org/10.

1016/0022-247X(85)90317-8. URL https://www.sciencedirect.com/science/article/pii/0022247X85903178.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://arxiv.org/abs/2012.09812.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017a.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017b. URL http://dx.doi.org/10.1038/nature24270.

David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2021.103535. URL https://www.sciencedirect.com/science/article/pii/S0004370221000862.

Anya Sims, Cong Lu, and Yee Whye Teh. The edge-of-reach problem in offline model-based reinforcement learning. In *Second Agent Learning in Open-Endedness Workshop*, 2023. URL https://openreview.net/forum?id=VjqNosdlAn.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007. doi: https://doi.org/10.1111/j.1467-7687.2007.00569.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-7687.2007.00569.x.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Yihao Sun, Jiaji Zhang, Chengxing Jia, Haoxin Lin, Junyin Ye, and Yang Yu. Model-Bellman inconsistency for model-based offline reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference*

on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pages 33177–33194. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sun23q.html.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Andrew Szot, Max Schwarzer, Bogdan Mazoure, Harsh Agrawal, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. Large language models as generalizable policies for embodied tasks. *preprint*, 2023.

Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion. *arXiv preprint arXiv:2305.11846*, 2023.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning, 2023.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm_control: Software and tasks for continuous control, 2020.

Emanuel Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1369–1376. MIT Press, 2007.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10295. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10295`.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad, 2018.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, Chelsea Finn,

Max Du, Moo Jin Kim, Alexander Khazatsky, Jonathan Heewon Yang, Tony Z. Zhao, Ken Goldberg, Ryan Hoque, Lawrence Yunliang Chen, Simeon Adebola, Gaurav S. Sukhatme, Gautam Salhotra, Shivin Dass, Lerrel Pinto, Zichen Jeff Cui, Siddhant Haldar, Anant Rai, Nur Muhammad Mahi Shafiullah, Yuke Zhu, Yifeng Zhu, Soroush Nasiriany, Shuran Song, Cheng Chi, Chuer Pan, Wolfram Burgard, Oier Mees, Chenguang Huang, Deepak Pathak, Shikhar Bahl, Russell Mendonca, Gaoyue Zhou, Mohan Kumar Srirama, Sudeep Dasari, Cewu Lu, Hao-Shu Fang, Hongjie Fang, Henrik I Christensen, Masayoshi Tomizuka, Wei Zhan, Mingyu Ding, Chenfeng Xu, Xinghao Zhu, Ran Tian, Youngwoon Lee, Dorsa Sadigh, Yuchen Cui, Suneel Belkhale, Priya Sundaresan, Trevor Darrell, Jitendra Malik, Ilija Radosavovic, Jeannette Bohg, Krishnan Srinivasan, Xiaolong Wang, Nicklas Hansen, Yueh-Hua Wu, Ge Yan, Hao Su, Jiayuan Gu, Xuanlin Li, Niko Suenderhauf, Krishan Rana, Ben Burgess-Limerick, Federico Ceola, Kento Kawaharazuka, Naoaki Kanazawa, Tatsuya Matsushima, Yutaka Matsuo, Yusuke Iwasawa, Hiroki Furuta, Jihoon Oh, Tatsuya Harada, Takayuki Osa, Yujin Tang, Oliver Kroemer, Mohit Sharma, Kevin Lee Zhang, Beomjoon Kim, Yoonyoung Cho, Junhyek Han, Jaehyung Kim, Joseph J Lim, Edward Johns, Norman Di Palo, Freek Stulp, Antonin Raffin, Samuel Bustamante, João Silvério, Abhishek Padalkar, Jan Peters, Bernhard Schölkopf, Dieter Büchler, Jan Schneider, Simon Guist, Jiajun Wu, Stephen Tian, Haochen Shi, Yunzhu Li, Yixuan Wang, Mingtong Zhang, Heni Ben Amor, Yifan Zhou, Keyvan Majd, Lionel Ott, Giulio Schiavi, Roberto Martín-Martín, Rutav Shah, Yonatan Bisk, Jeffrey T Bingham, Tianhe Yu, Vidhi Jain, Ted Xiao, Karol Hausman, Christine Chan, Alexander Herzog, Zhuo Xu, Sean Kirmani, Vincent Vanhoucke, Ryan Julian, Lisa Lee, Tianli Ding, Yevgen Chebotar, Jie Tan, Jacky Liang, Igor Mordatch, Kanishka Rao, Yao Lu, Keerthana Gopalakrishnan, Stefan Welker, Nikhil J Joshi, Coline Manon Devin, Alex Irpan, Sherry Moore, Ayzaan Wahid, Jialin Wu, Xi Chen, Paul Wohlhart, Alex Bewley, Wenxuan Zhou, Isabel Leal, Dmitry Kalashnikov, Pannag R Sanketi, Chuyuan Fu, Ying Xu, Sichun Xu, brian ichter, Jasmine Hsu, Peng Xu, Anthony Brohan, Pierre Sermanet, Nicolas

Heess, Michael Ahn, Rafael Rafailov, Acorn Pooley, Kendra Byrne, Todor Davchev, Kenneth Oslund, Stefan Schaal, Ajinkya Jain, Keegan Go, Fei Xia, Jonathan Tompson, Travis Armstrong, and Danny Driess. Open x-embodiment: Robotic learning datasets and RT-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023*, 2023. URL `https://openreview.net/forum?id=zraBtFgxT0`.

Fritz R Walther. Flight behaviour and avoidance of predators in thomson's gazelle (gazella thomsoni guenther 1884). *Behaviour*, pages 184–221, 1969.

M. Waltz and K. Fu. A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, 10(4):390–398, 1965. doi: 10.1109/TAC.1965.1098193.

Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. In *ICML*, 2021.

Xingchen Wan, Cong Lu, Jack Parker-Holder, Philip J. Ball, Vu Nguyen, Binxin Ru, and Michael Osborne. Bayesian Generational Population-Based Training. In *AutoML*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran

Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/01ce84968c6969bdd5d51c5eeaa3946a-Paper.pdf.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9940–9951. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/wang20l.html.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8: 279–292, 1992.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=3RBY8fKjHeu.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, Zsolt Kira, Manolis Savva, Angel Chang, Devendra Singh Chaplot, Dhruv Batra, Roozbeh Mottaghi, Yonatan Bisk, and Chris Paxton. Homerobot: Open-vocabulary mobile manipulation, 2023.

Kenny John Young, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. The benefits of model-based generalization in reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato,

and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 40254–40276. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/young23a.html.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020a. doi: 10.1109/cvpr42600.2020.00271.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf.

Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Brian Ichter, Karol Hausman, and Fei Xia. Scaling robot learning with semantically imagined experience. In *arXiv preprint arXiv:2302.11550*, 2023.

Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

Hongyu Zang, Xin Li, Jie Yu, Chen Liu, Riashat Islam, Remi Tachet Des Combes, and Romain Laroche. Behavior prior representation learning for offline reinforcement learning. *arXiv preprint arXiv:2211.00863*, 2022.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Learning robust state abstractions for hidden-parameter block MDPs. In *International Conference on Learning Representations*, 2021a.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021b.

Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. Omni: Open-endedness via models of human notions of interestingness, 2023a.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023b.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned ai. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15763–15773. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/b607ba543ad05417b8507ee86c54fcb7-Paper.pdf`.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1433–1438. AAAI Press, 2008. ISBN 9781577353683.

Luisa Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in Approximate Hyper-State Space for Meta Reinforcement Learning. In *ICML*, 2021a.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational

bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021b. URL `http://jmlr.org/papers/v22/21-0657.html`.

Appendices

A

Appendices of Chapter 3

Appendix

A. Additional Experiments

In this section we show the performance for Augmented World Models with different training ranges for the DAS augmentation (z train in Table 4). We train with adaptive context on the HalfCheetah mixed dataset, and present the results in Fig. 13. As we see, $[0.75, 1.25]$ and $[0.5, 1.5]$ perform the best. Based on this, we use $[0.5, 1.5]$ for our experiments as we believe this helps us sample a wider set of dynamics, helping us generalize better across all environments and data sets.

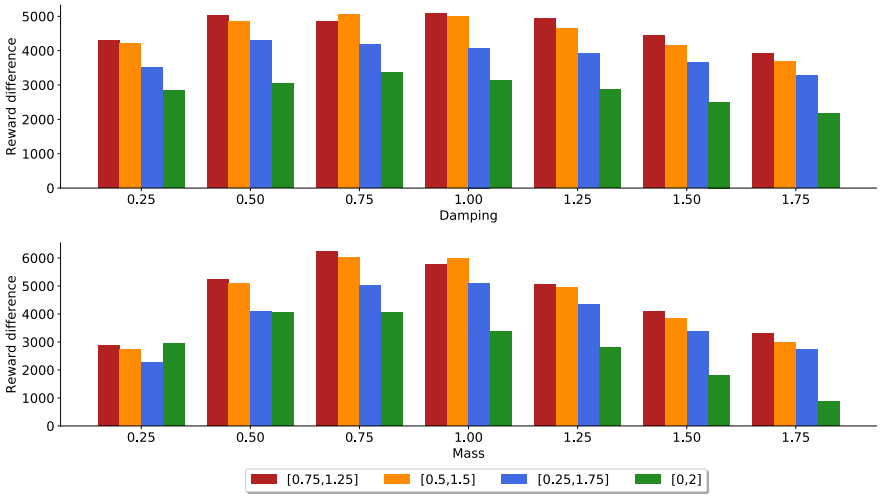


Figure 13. Performance for Augmented World Models with the DAS augmentation. Each plot shows different values for a and b, the ranges for the sampled noise.

B. Implementation Details

B.1. Hyperparameters

Our algorithm is based on MOPO Yu et al. (2020) with values for the rollout length h and penalty coefficient λ shown in Table 3.

Table 3. Hyperparameters used in the D4RL datasets.

Dataset Type	**Environment**	**MOPO (h, λ)**
random	halfcheetah	5, 0.5
random	walker2d	1, 1
medium	halfcheetah	1, 1
medium	walker2d	1, 1 [4]
mixed	halfcheetah	5, 1
mixed	walker2d	1, 1
med-expert	halfcheetah	5, 1
med-expert	walker2d	1, 2

AugWM specific hyperparameters are listed in Table 4. For each evaluation rollout, we clear the buffer of stored true modified environment transitions to measure zero-shot performance. We adapt using the context after a set number of steps, k, in the environment to train the linear model. The two ranges used for the context z during training and test time are different. At test time, the estimated context is clipped to remain within the given bounds.

[4]We follow the original MOPO hyperparameters for all datasets except for walker2d-medium where we found (1, 1) worked better for both MOPO and our method than (5, 5).

Table 4. AugWM Hyperparameters

Parameter	Value
evaluation rollouts	5
MOPO offline epochs	400
AugWM offline epochs	900
k, steps for adaptation	300
z train range	[0.5, 1.5]
z test range	[0.93, 1.07]

B.2. D4RL dataset

We evaluate our method on D4RL Fu et al. (2021) datasets based on the MuJoCo continuous control tasks (halfcheetah and walker2d). The four dataset types we evaluate on are:

- **random:** roll out a randomly initialized policy for 1M steps.
- **medium:** partially train a policy using SAC, then roll it out for 1M steps.
- **mixed:** train a policy using SAC until a certain (environment-specific) performance threshold is reached, and take the replay buffer as the batch.
- **medium-expert:** combine 1M samples of rollouts from a fully-trained policy with another 1M samples of rollouts from a partially trained policy or a random policy.

This gives us a total of 8 experiments.

B.3. Ant Environment

For the Ant experiments, we follow the Ant Changed Direction approach in MOPO Yu et al. (2020). Since this offline dataset is not provided in the authors' code, nor is it in the standard D4RL library Fu et al. (2021), we were required to generate our own offline Ant dataset. Since the authors' did not outline certain details in their experiment, we found the following was required to match their performance with our codebase: 1) Training our SAC policy for 1×10^6 timesteps in the Ant environment provided by the authors' code in Yu et al. (2020); 2) relabelling each reward in the buffer using the new direction, without the living reward; 3) training a world model over this offline dataset; 4) training a policy in the world model, adding in living reward post-hoc; 5) evaluating the policy with the living reward.

B.4. HalfCheetah Modified Agent

We use the modified HalfCheetah environments from Henderson et al. (2017). In each setting one body part of the agent is changed, from following set: {Foot, Leg, Thigh, Torso, Head}. The body part can either be "Big" or "Small", where Big bodyparts involve scaling the mass and width of the limb by 1.25 and Small bodyparts are scaled by 0.75. In Table 2 we show the mean over each of these five body parts, for agents trained on each of the D4RL datasets, repeated for five seeds.

B

Appendices of Chapter 4

Supplementary Material

A Data Modeling

In this section, we provide further details for our data modeling. Our diffusion model generates full environment transitions i.e., a concatenation of states, actions, rewards, next states, and terminals where they are present. For the purposes of modeling, we normalize each continuous dimension (non-terminal) to have 0 mean and 1 std. We visualize the marginal distributions over the state, action, and reward dimensions on the standard halfcheetah medium-replay dataset in Figure 8 and observe that the synthetic samples accurately match the high-level statistics of the original dataset.

We note the difficulties of appropriately modeling the terminal variable which is a binary variable compared to the rest of the dimensions which are continuous for the environments we investigate. This is particularly challenging for "expert" datasets where early termination is rare. For example, walker2d-expert only has $\approx 0.0001\%$ terminals. In practice, we find it sufficient to leave the terminals un-normalized and round them to 0 or 1 by thresholding the continuous diffusion samples in the middle at 0.5. A cleaner treatment of this variable could be achieved by leveraging work on diffusion with categorical variables [31].

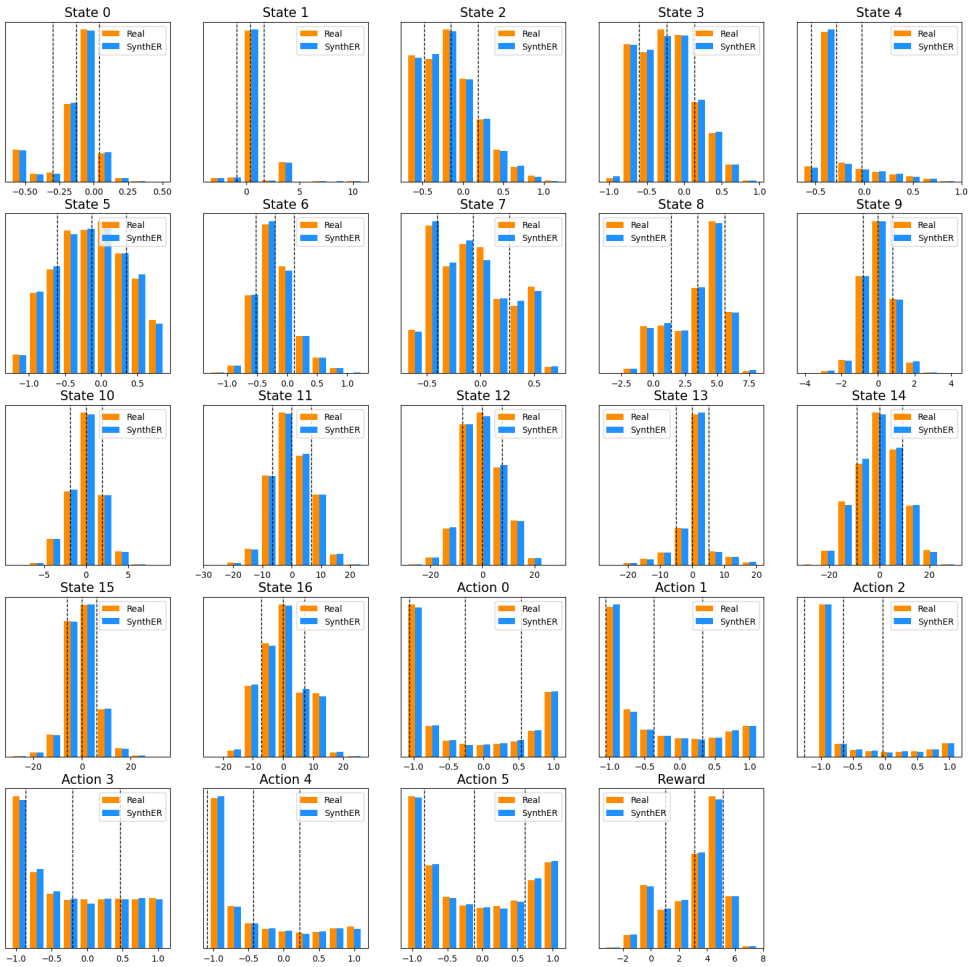


Figure 8: Histograms of the empirical marginal distribution of samples from SYNTHER in **blue** on the halfcheetah medium-replay dataset against the original data in **orange**. Dashed lines indicate the mean $\pm$ one standard deviation in the original dataset. SYNTHER faithfully reproduces the high-level statistics of the dataset.

A.1 Data Compression

An immediate advantage of sampling data from a generative model is compression. In Table 5, we compare the memory requirements of SYNTHER and the original data by the number of 32-bit floating point numbers used by each for some sample D4RL [21] datasets. For the original data, this simply scales linearly with the size of the dataset. On other hand, SYNTHER amortizes this in the number of parameters in the denoising network, resulting in a high level of dataset compression, at the cost of sampling speed. This property was also noted in the continual learning literature with generative models summarizing previous tasks [65]. As we discuss in Appendix B.3, sampling is fast with 100K transitions taking around 90 seconds to generate.

Table 5: SYNTHER provides high levels of dataset compression *without sacrificing downstream performance* in offline reinforcement learning. Statistics shown are for the standard D4RL MuJoCo walker2d datasets which has a transition dimension of 42, and the residual denoiser used for evaluation on these environments in Section 4.1. Figures are given to 1 decimal place.

Dataset	# FP32s in Original Dataset	# Diffusion Parameters	Compression
mixed	12.6M	6.5M	1.9×
medium	42M		6.5×
medium-expert	84M		12.9×

B Hyperparameters

B.1 TVAE and CTGAN

In Section 3.1, we compared SYNTHER to the VAE and GAN baselines, TVAE and CTGAN. As these algorithms have not been used for reinforcement learning data before, we performed a hyperparameter search [42] across the following spaces:

Table 6: Hyperparameter search space for TVAE. We highlight the default choice in **bold**.

Parameter	Search Space
no. layers	{ 1, **2**, 3, 4 }
width	{ 64, **128**, 256, 512 }
batch size	{ 250, **500**, 1000 }
embedding dim	{ 32, 64, **128**, 256 }
loss factor	{ 0.02, 0.2, **2**, 20}

Table 7: Hyperparameter search space for CTGAN. We highlight the default choice in **bold**.

Parameter	Search Space
no. layers	{ 1, **2**, 3, 4 }
width	{ 64, 128, **256**, 512 }
batch size	{ 250, **500**, 1000 }
embedding dim	{ 32, 64, **128**, 256 }
discriminator steps	{ **1**, 2}

These ranges are similar to those listed in Tables 10 and 11 of Kotelnikov et al. [42]. We used 30 trials along with the default.

B.2 Denoising Network

The formulation of diffusion we use in our paper is the Elucidated Diffusion Model (EDM, Karras et al. [38]). We parametrize the denoising network D_θ as an MLP with skip connections from the previous layer as in Tolstikhin et al. [70]. Thus each layer has the form given in Equation (3).

$$x_{L+1} = \text{linear}(\text{activation}(x_L)) + x_L \tag{3}$$

The hyperparameters are listed in Table 8. The noise level of the diffusion process is encoded by a Random Fourier Feature [57] embedding. The base size of the network uses a width of 1024 and

depth of 6 and thus has ≈ 6M parameters. We adjust the batch size for training based on dataset size. For online training and offline datasets with fewer than 1 million samples (medium-replay datasets) we use a batch size of 256, and 1024 otherwise.

For the following offline datasets, we observe more performant samples by increasing the width up to 2048: halfcheetah medium-expert, hopper medium, and hopper medium-expert. This raises the network parameters to ≈ 25M, which remains fewer parameters than the original data as in Table 5. We provide ablations on the depth and type of network used in Table 10.

Table 8: Default Residual MLP Denoiser Hyperparameters.

Parameter	Value(s)
no. layers	6
width	1024
batch size	{ 256 for online and medium-replay, 1024 otherwise }
RFF dimension	16
activation	relu
optimizer	Adam
learning rate	3×10^{-4}
learning rate schedule	cosine annealing
model training steps	100K

B.3 Elucidated Diffusion Model

For the diffusion sampling process, we use the stochastic SDE sampler of Karras et al. [38] with the default hyperparameters used for the ImageNet, given in Table 9. We use a higher number of diffusion timesteps at 128 for improved sample fidelity. We use the implementation at `https://github.com/lucidrains/denoising-diffusion-pytorch` which is released under an Apache license.

Table 9: Default ImageNet-64 EDM Hyperparameters.

Parameter	Value
no. diffusion steps	128
σ_{min}	0.002
σ_{max}	80
S_{churn}	80
S_{tmin}	0.05
S_{tmax}	50
S_{noise}	1.003

The diffusion model is fast to train, taking approximately 17 minutes for 100K training steps on a standard V100 GPU. It takes approximately 90 seconds to generate 100K samples with 128 diffusion timesteps.

C SYNTHER Ablations

We consider ablations on the number of generated samples and type of denoiser used for our offline evaluation in Section 4.1.

C.1 Size of Upsampled Dataset

In our main offline evaluation in Section 4.1, we upsample each dataset (which has an original size of between 100K to 2M) to 5M. We investigate this choice for the walker medium-replay dataset in Figure 9 and choose 10 levels log-uniformly from the range [50K, 5M]. Similarly to He et al. [26], we find that performance gains with synthetic data eventually saturate and that 5M is a reasonable heuristic for all our offline datasets.

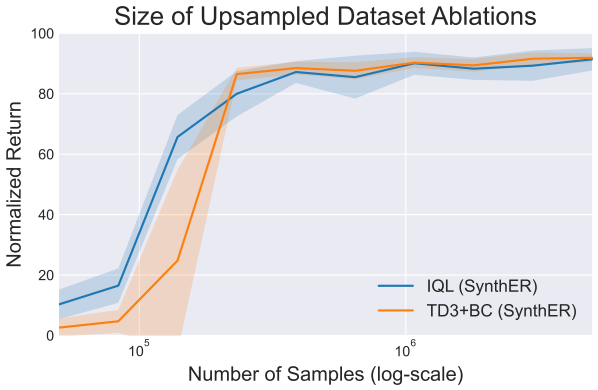


Figure 9: Ablations on the number of samples generated by SYNTHER for the offline walker medium-replay dataset. We choose 10 levels log-uniformly from the range [50K, 5M]. We find that performance eventually saturates at around 5M samples.

C.2 Network Ablations

We ablate the hyperparameters of the denoising network, comparing 3 settings of depth from $\{2, 4, 6\}$ and analyze the importance of skip connections. The remaining hyperparameters follow Appendix B.2. We choose the hopper medium-expert dataset as it is a large dataset of 2M. As we can see in Table 10, we see a positive benefit from the increased depth and skip connections which leads to our final choice in Table 8.

Table 10: Ablations on the denoiser network used for SYNTHER on the hopper medium-expert dataset. We observe that greater depth and residual connections are beneficial for downstream offline RL performance. We show the mean and standard deviation of the final performance averaged over 4 seeds.

Network	Depth	Eval. Return
MLP	2	86.8±18.7
	4	89.9±17.9
	6	**100.4± 6.9**
Residual MLP	2	78.5±11.3
	4	**99.3±14.7**
	6	**101.1±10.5**

D RL Implementation

For the algorithms in the offline RL evaluation in Section 4.1, we use the 'Clean Offline Reinforcement Learning' (CORL, Tarasov et al. [68]) codebase. We take the final performance they report for the baseline offline evaluation. Their code can be found at `https://github.com/tinkoff-ai/CORL` and is released under an Apache license.

For the online evaluation, we consider Soft Actor-Critic (SAC, Haarnoja et al. [24]) and use the implementation from the REDQ [12] codebase. This may be found at `https://github.com/watchernyu/REDQ` and is released under an MIT license. We use the 'dmcgym' wrapper for the DeepMind Control Suite [71]. This may be found at `https://github.com/ikostrikov/dmcgym` and is released under an MIT license.

D.1 Data Augmentation Hyperparameters

For the data augmentation schemes we visualize in Figure 1a, we define:

1. Additive Noise [45]: adding $\epsilon \sim \mathcal{N}(0, 0.1)$ to s_t and s_{t+1}.
2. Multiplicative Noise [45]: multiplying s_t and s_{t+1} by single number $\epsilon \sim \text{Unif}([0.8, 1.2])$.

3. Dynamics Noise [8]: multiplying the next state delta $s_{t+1} - s_t$ by $\epsilon \sim \text{Unif}([0.5, 1.5])$ so that $s_{t+1} = s_t + \epsilon \cdot (s_{t+1} - s_t)$.

D.2 Online Running Times

Our online implementation in Section 4.2 uses the default training hyperparameters in Appendix B.2 to train the diffusion model every 10K online steps, and generates 1M transitions each time. On the 200K DMC experiments, 'SAC (SynthER)' takes ≈ 21.1 hours compared to ≈ 22.7 hours with REDQ on a V100 GPU. We can further break down the running times of 'SAC (SynthER)' as follows:

- Diffusion training: 4.3 hours
- Diffusion sampling: 5 hours
- RL training: 11.8 hours

Therefore, the majority of training time is from reinforcement learning with an update-to-data ratio (UTD) of 20. We expect the diffusion training may be heavily sped-up with early stopping, and leave this to future work. The default SAC algorithm with UTD=1 takes ≈ 2 hours.

E Further Offline Results

In this section, we include additional supplementary offline experiments to those presented in Section 4.1.

E.1 AntMaze Data Generation

We further verify that SYNTHER can generate synthetic data for more complex environments such as AntMaze [21]. This environment replaces the 2D ball from Maze2D with the more complex 8-DoF "Ant" quadruped robot, and features: non-Markovian policies, sparse rewards, and multitask data. In Table 11, we see that SYNTHER improves the TD3+BC algorithm where it trains (on the 'umaze' dataset) and achieves parity otherwise.

Table 11: We show synthetic data from SYNTHER achieves at least parity for more complex offline environments like AntMaze-v2, evaluated with the TD3+BC algorithm. We show the mean and standard deviation of the final performance averaged over 6 seeds.

Environment		**TD3+BC [22]**	
		Original	**SynthER**
AntMaze	umaze	70.8±39.2	**88.9±4.4**
	medium-play	0.3±0.4	0.5±0.7
	large-play	0.0±0.0	0.0±0.0

E.2 Offline Data Mixtures

In Section 4.1, we considered exclusively training on synthetic data. We present results in Table 12 with a 50-50 mix of real and synthetic data to confirm that the two are compatible with each other, similar to Section 4.2. We do so by including as many synthetic samples as there are real data. As we stated before, we do not expect an increase in performance here due to the fact that most D4RL datasets are at least 1M in size and are already sufficiently large.

Table 12: We verify that the synthetic data from SYNTHER can be mixed with the real data for offline evaluation. The 50-50 mix achieves parity with the original data, same as the synthetic data. We show the mean and standard deviation of the final performance averaged over 8 seeds.

Environment	**TD3+BC [22]**			**IQL [41]**		
	Original	**SYNTHER**	**50-50**	**Original**	**SYNTHER**	**50-50**
locomotion average	59.0±4.9	60.0±5.1	59.2±3.7	62.1±3.5	63.7±3.5	62.7±5.1

F Latent Data Generation with V-D4RL

We provide full details for the experiments in Section 4.3 that scale SYNTHER to pixel-based environments by generating data in latent space for the DrQ+BC [50] and BC algorithms. Concretely, for the DrQ+BC algorithm, we consider parametric networks for the shared CNN encoder, policy, and Q-functions, f_ξ, π_ϕ, and Q_θ respectively. We also use a random shifts image augmentation, aug. Therefore, the Q-value for a state s and action a is given by $Q_\theta(f_\xi(\text{aug}(s)), a)$. The policy is similarly conditioned on an encoding of an augmented image observation.

The policy and Q-functions both consist of an initial 'trunk' which further reduces the dimensionality of the CNN encoding to $d_{\text{feature}} = 50$, followed by fully connected layers. We represent this as $\pi_\phi = \pi_\phi^{\text{fc}} \circ \pi_\phi^{\text{trunk}}$ and $Q_\theta = Q_\theta^{\text{fc}} \circ Q_\theta^{\text{trunk}}$. This allows us to reduce a pixel-based transition to a low-dimensional latent version. Consider a pixel-based transition (s, a, r, s') where $s, s' \in \mathbb{R}^{84\times84\times3}$. Let $h = f_\xi(\text{aug}(s))$ and $h' = f_\xi(\text{aug}(s'))$. The latent transition we generate is:

$$(\pi_\phi^{\text{trunk}}(h), Q_\theta^{\text{trunk}}(h), a, r, \pi_\phi^{\text{trunk}}(h'), Q_\theta^{\text{trunk}}(h'))$$

This has dimension $4 \cdot d_{\text{feature}} + |a| + 1$ and includes specific supervised features for both the actor and the critic; we analyze this choice in Appendix F.1. For example, on the 'cheetah-run' environment considered in V-D4RL, since $|a| = 6$, the overall dimension is 207 which is suitable for our residual MLP denoising networks using the same hyperparameters in Table 8. This allows us to retain the fast training and sampling speed from the proprioceptive setting but now in pixel space.

To obtain a frozen encoder f_ξ and trunks π_ϕ^{trunk}, Q_θ^{trunk}, we simply train in two stages. The first stage trains the original algorithm on the original data. The second stage then retrains only the fully-connected portions of the actor and critic, π_ϕ^{fc} and Q_θ^{fc}, with synthetic data. Thus, our approach could also be viewed as fine-tuning the heads of the networks. The procedure for the BC algorithm works the same but without the critic.

We use the official V-D4RL [50] codebase for the data and algorithms in this evaluation. Their code can be found at `https://github.com/conglu1997/v-d4rl` and is released under an MIT license.

F.1 Ablations On Representation

We analyze the choice of low-dimensional latent representation we use in the previous section, in particular, using specific supervised features for both the actor and critic. We compare this against using actor-only or critic-only features for both the actor and critic, which corresponds to a choice of $\pi_\phi^{\text{trunk}} = Q_\theta^{\text{trunk}}$, in Table 13. We note that both perform worse with an especially large drop-off for the critic-only features. This may suggest that non-specific options for compressing the image into low-dimensional latents, for example, using auto-encoders [40], could be even less suitable for this task.

Table 13: Ablations on the latent representation used for SYNTHER on the V-D4RL cheetah expert dataset. We observe that separate specific supervised features are essential for downstream performance with a particularly large decrease if we only used critic features for the actor and critic. We show the mean and standard deviation of the final performance averaged over 4 seeds.

Latent Representation	Eval. Return
Actor and Critic (Ours)	**52.3±7.0**
Actor Only	43.5±7.3
Critic Only	16.0±2.8

C

Appendices of Chapter 5

Supplementary Material

Table of Contents

Appendix A Derivations and Further Technical Details

A.1 Proof of Proposition 1

Proposition 1 (Exploding Gradients in KL-Regularized RL)**.** *Let $\pi_0(\cdot \,|\, \mathbf{s})$ be a Gaussian behavioral reference policy with mean $\boldsymbol{\mu}_0(\mathbf{s})$ and variance $\boldsymbol{\sigma}_0^2(\mathbf{s})$, and let $\pi(\cdot \,|\, \mathbf{s})$ be an online policy with reparameterization $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$ and random vector ϵ_t. The gradient of the policy loss with respect to the online policy's parameters ϕ is then given by*

$$\begin{aligned}\hat{\nabla}_\phi J_\pi(\phi) = \big(&\alpha \nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t \,|\, \mathbf{s}_t) - \alpha \nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) \\ &- \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)\big) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) + \alpha \nabla_\phi \log \pi_\phi(\mathbf{a}_t \,|\, \mathbf{s}_t)\end{aligned} \tag{A.1}$$

with

$$\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) = -\frac{\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)}{\boldsymbol{\sigma}_0^2(\mathbf{s}_t)}. \tag{A.2}$$

For fixed $|\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)|$, $\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t)$ grows as $\mathcal{O}(\boldsymbol{\sigma}_0^{-2}(\mathbf{s}_t))$; thus,

$$|\, \hat{\nabla}_\phi J_\pi(\phi) \,| \to \infty \quad \textit{as} \quad \boldsymbol{\sigma}_0^2(\mathbf{s}_t) \to 0, \tag{A.3}$$

when $\nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \neq 0$.

Proof. The policy loss, as given in Equation (3), is:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\mathbb{D}_{\text{KL}}\big(\pi_\phi(\cdot \,|\, \mathbf{s}_t) \,||\, \pi_0(\cdot \,|\, \mathbf{s}_t)\big)\right] - \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right]\right]. \tag{A.4}$$

To obtain a lower-variance gradient estimator, the policy is reparameterized using a neural network transformation

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t) \tag{A.5}$$

where ϵ_t is an input noise vector. Following Haarnoja et al. [13], we can now rewrite Equation (A.4) as

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t}\left[\alpha\big(\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t) \,|\, \mathbf{s}_t) - \log \pi_0(f_\phi(\epsilon_t; \mathbf{s}_t) \,|\, \mathbf{s}_t)\big) - Q(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t))\right] \tag{A.6}$$

where $\mathcal{D}$ is a replay buffer and π_ϕ is defined implicitly in terms of f_ϕ. We can approximate the gradient of Equation (A.6) with

$$\begin{aligned}\hat{\nabla}_\phi J_\pi(\phi) = \big(&\alpha \nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t \,|\, \mathbf{s}_t) - \alpha \nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) \\ &- \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)\big) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) + \alpha \nabla_\phi \log \pi_\phi(\mathbf{a}_t \,|\, \mathbf{s}_t).\end{aligned} \tag{A.7}$$

Next, consider the term $\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t)$ for a Gaussian policy:

$$\log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) = \log\left(\frac{1}{\boldsymbol{\sigma}_0(\mathbf{s}_t)\sqrt{2\pi}}\right) - \frac{1}{2}\left(\frac{\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)}{\boldsymbol{\sigma}_0(\mathbf{s}_t)(\mathbf{s}_t)}\right)^2 \tag{A.8}$$

Thus,

$$\nabla_{\mathbf{a}_t} \log \pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) = -\frac{\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)}{\boldsymbol{\sigma}_0^2(\mathbf{s}_t)}. \tag{A.9}$$

For fixed $|\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)|$, $\nabla_{\mathbf{a}_t} \log(\pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t))$ grows as $\mathcal{O}(\boldsymbol{\sigma}_0^{-2}(\mathbf{s}_t))$, and so,

$$|\hat{\nabla}_\phi J_\pi(\phi)| \to \infty \quad \text{as} \quad \boldsymbol{\sigma}_0^2(\mathbf{s}_t) \to 0. \tag{A.10}$$

whenever $\nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \neq 0$. □

A.2 Laplace Parametric Behavioral Reference Policy

A Laplace behavioral reference policy may be able to mitigate some of the problems posed by Proposition 1 due to the heavy tails of the distribution. The gradient for a Laplace behavioral reference policy

$$\pi_0(\mathbf{a}_t \,|\, \mathbf{s}_t) \doteq \frac{1}{2\boldsymbol{\sigma}_0(\mathbf{s}_t)} \exp\left(-\frac{|\mathbf{a}_t - \boldsymbol{\mu}_0(\mathbf{s}_t)|}{\boldsymbol{\sigma}_0(\mathbf{s}_t)}\right), \tag{A.11}$$

increases linearly for a given distance between $\mathbf{a}_t$ and the mean $\boldsymbol{\mu}_0(\mathbf{s}_t)$ as the scale $\boldsymbol{\sigma}_0(\mathbf{s}_t)$ tends to zero.

A.3 Regularized Maximum Likelihood Estimation

To address the collapse in predictive variance away from the offline dataset under MLE training seen in Figure 1, Wu et al. [51] in practice augment the usual MLE loss with an entropy bonus as follows:

$$\pi_0 \doteq \pi_{\psi^\star} \quad \text{with} \quad \psi^\star \doteq \arg\max_\psi \left\{\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}[\log \pi_\psi(\mathbf{a} \,|\, \mathbf{s}) + \beta \mathcal{H}(\pi_\psi(\cdot \,|\, \mathbf{s}))]\right\}. \tag{A.12}$$

where β is temperature tuned to an entropy constraint similar to Haarnoja et al. [13]. The entropy bonus is estimated by sampling from the behavioral policy as

$$\mathcal{H}(\pi_\psi(\cdot \,|\, \mathbf{s})) = \mathbb{E}_{\mathbf{a}\sim\pi_\psi}[-\log \pi_\psi(\mathbf{a} \,|\, \mathbf{s})] \tag{A.13}$$

Figure 11 shows the predictive variances of behavioral policies trained on expert demonstrations for the "door-binary-v0" environment with various entropy coefficients β. Whilst entropy regularization partially mitigates the collapse of predictive variance away from the expert demonstrations, we still observe the wrong trend similar to Figure 1 with predictive variances high near the expert demonstrations and low on unseen data. The variance surface also becomes more poorly behaved, with "islands" of high predictive variance appearing away from the data.

We may also add Tikhonov regularization [12] to the MLE objective, explicitly,

$$\pi_0 \doteq \pi_{\psi^\star} \quad \text{with} \quad \psi^\star \doteq \arg\max_\psi \left\{\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}[\log \pi_\psi(\mathbf{a} \,|\, \mathbf{s}) - \lambda \psi^\top \psi]\right\}. \tag{A.14}$$

where λ is the regularization coefficient.

Figure 12 shows the predictive variances of behavioral policies trained on expert demonstrations for the "door-binary-v0" environment with varying Tikhonov regularization coefficients λ. Similarly, Tikhonov regularization does not resolve the issue with calibration of uncertainties. We also observe that too high a regularization strength causes the model to underfit to the variances of the data.

A.4 Comparison to Prior Works

To assess the usefulness of KL regularization for improving the performance and sample efficiency of online learning with expert demonstrations, we compare our approach to methods that incorporate expert demonstrations into online learning implicitly or explicitly via KL regularization as well as by means other than KL regularization.

ABM [44]. ABM explicitly KL-regularizes the online policy against a behavioral policy. This behavioral policy can be estimated via MLE, like BRAC, or alternatively via an "advantage-weighted behavioral model" where the RL algorithm is biased to choose actions that are both supported by the offline data and that are good for the current task. This objective filters trajectory snippets by advantage-weighting, using an n-step advantage function. We show that no carefully chosen objective with additional hyperparameters is required.

AWAC [27]. AWAC performs online fine-tuning of a policy pre-trained on offline. It achieves state-of-the-art results on the dexterous hand manipulation and MuJoCo continuous locomotion tasks. AWAC implicitly constrains the KL divergence of the online policy to be close to the behavioral policy by sampling from the replay buffer, which is initially filled with the offline data. The method requires additional off-policy data to be generated to saturate the replay buffer, thereby requiring a hidden number of environment interactions that do not involve learning. Our approach does not require the offline data to be added to the replay buffer before training.

AWR [31]. AWR approximates constrained policy search by alternating between supervised value function and policy regression steps. The objective derived is similar to AWAC but instead estimates the value function of the behavioral policy which was demonstrated to be less efficient than Q-function estimation via bootstrapping [27]. The method may be converted to use offline data by adding prior data to the replay buffer before training.

BEAR [19]. BEAR attempts to stabilize learning from off-policy data (such as offline data) by tackling bootstrapping error from actions far from the training data. This is achieved by searching for policies with the same support as the training distribution. This approach is too restrictive for the problem considered in this paper, since only a small number of expert demonstrations is available, which requires exploration. In contrast, our approach encourages exploration away from the data by wider behavioral policy predictive variances. BEAR uses an alternate divergence measure to the KL divergence, Maximum Mean Discrepancy [45]. Other divergences such as Wasserstein Distances [30] have also been proposed for regularization in RL.

BRAC [51]. BRAC regularizes the online policy against an offline behavioral policy as our method does. However, BRAC exhibits the pathologies we have shown by learning a poor behavioral policy via MLE. To mitigate this, in practice, BRAC adds an entropy bonus to the supervised learning objective which stabilizes the variance around the training set but has no guarantees away from the data. We demonstrate that behavioral policy obtained via maximum likelihood estimation with entropy regularization exhibit a collapse in predictive uncertainty estimates way from the training data, resulting in the pathology described in Proposition 1.

DAPG [34]. DAPG incorporates offline data into policy gradients by initially pre-training with a behaviorally cloned policy and then augmenting the RL loss with a supervised-learning loss. We similarly pre-train the online policy at the start to avoid noisy KLs at the beginning of training. However, training a joint loss that combines two disparate and often divergent terms can be unstable.

SAC+BC [26]. SAC+BC represents the approach of Nair et al. [26] but uses SAC instead of DDPG [22] as the underlying RL algorithm. The method maintains a secondary replay buffer filled with offline data that is sampled each update step, augmenting the policy loss with a supervised learning loss that is filtered by advantage and hindsight experience replay. Our method requires far fewer additional ad-hoc algorithmic design choices.

SACfD [13]. SACfD uses the popular Soft Actor–Critic (SAC) algorithm with offline data loaded into the replay buffer before online training. Our algorithm uses the same approximate policy iteration scheme as SAC with a modified objective. Nair et al. [27] show that including the offline data into the replay buffer does not significantly improve the training performance over the unmodified SAC objective and that pre-training the online policy with offline data results in catastrophic forgetting. Thus, a different approach is needed to integrate offline data with SAC-style algorithms.

Appendix B Further Experimental Results

B.1 Exploding Q-function Gradients

In Proposition 1 and Section 3.4, we showed that the policy gradient $\hat{\nabla}_\phi J_\pi(\phi)$ explodes due to the blow-up of the gradient of the behavioral reference policy's log-density as the behavioral policy predictive variance $\boldsymbol{\sigma}_0(\mathbf{s})$ tends to zero. A similar relationship holds for the Q-function gradients, which we confirm empirically in Figure 8.

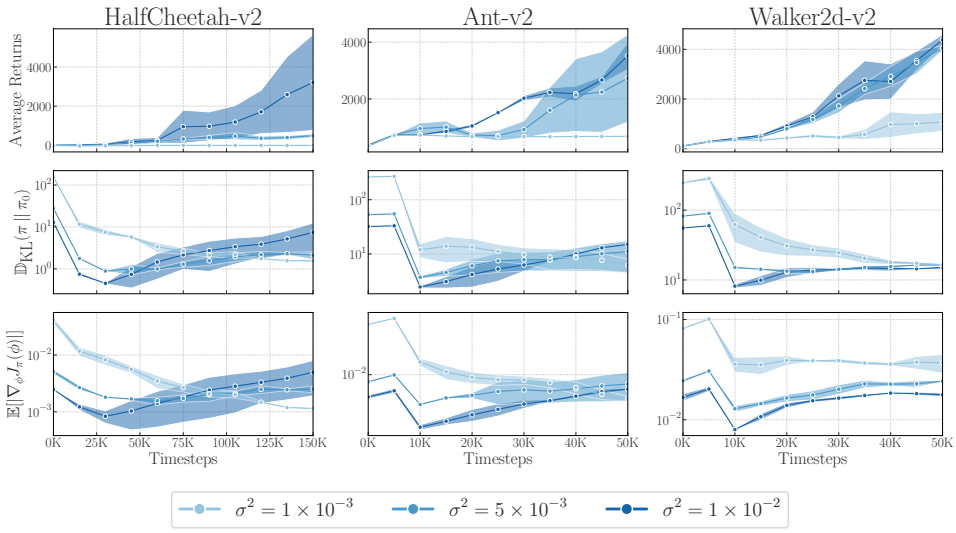


Figure 8: Ablation study showing the effect of predictive variance collapse on the performance of KL-regularized RL on MuJoCo benchmarks. Policies shown from dark to light in order of decreasing constant predictive variance, simulating training under maximum likelihood estimation. The plots show the average return of the learned policy, magnitude of the KL penalty, and magnitude of the Q-function gradients during online training.

B.2 Ablation Study on the Effect of KL Divergence Temperature Tuning

Figure 9 shows that unlike in standard SAC [13], tuning of the KL-temperature is not necessary to achieve good online performance. For simplicity, we use a fixed value throughout our experiments.

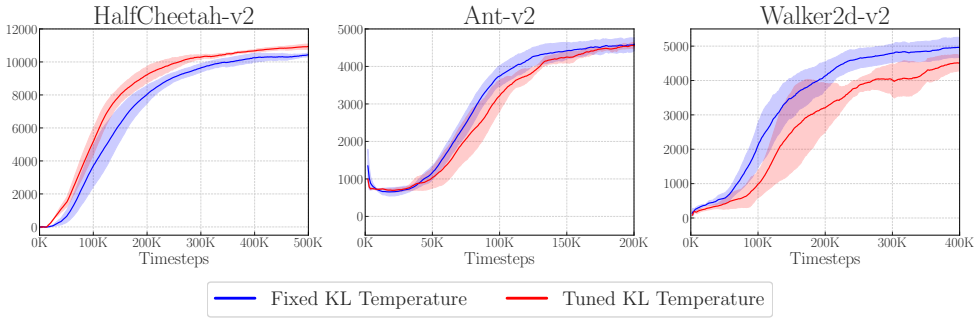


Figure 9: Ablation study on the effect of automatic KL temperature tuning on the performance of KL-regularized RL with a non-parametric GP behavioral reference policy on MuJoCo locomotion tasks.

B.3 Ablation Study: Performance under a Laplace Parametric Behavioral Reference Policy

We use a Laplace behavioral reference policy to assess whether it is more effective at incorporating the expert demonstration data into online training. Figure 10 shows empirical results using the Laplace behavioral reference policy compared against N-PPAC (in blue) and a SAC baseline (in green) on three MuJoCo locomotion tasks. We use automatic KL-temperature tuning for this ablation. On the Ant-v2 environment, the Laplace behavioral reference policy slightly improves upon the baseline SAC performance, which does not use any prior information at all. On the door and pen environment, the online policy learned under the Laplace behavioral reference policy does not learn any meaningful behavior.

In both MuJoCo locomotion tasks and the "door-binary-v0" and "pen-binary-v0" dexterous hand manipulation environments, N-NPAC significantly outperforms both the online policy learned under the Laplace behavioral reference policy and the SAC baseline. We can understand the behavior under the Laplace behavioral reference policy in terms of collapse of predictive variance away from data for neural network parameterized policies, as it too has a decreasing variance away from the expert trajectories.

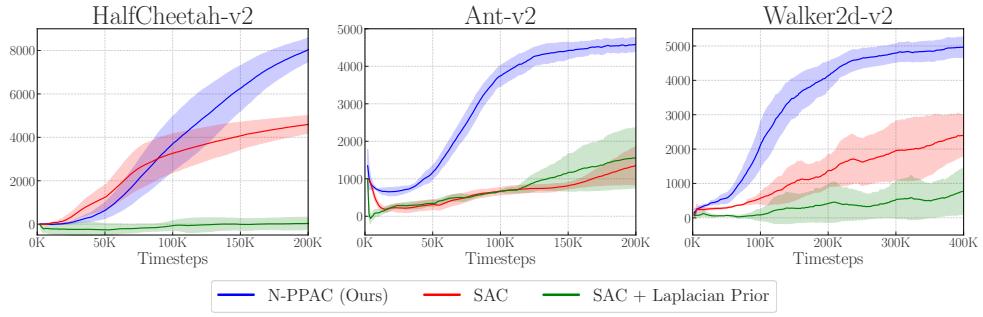
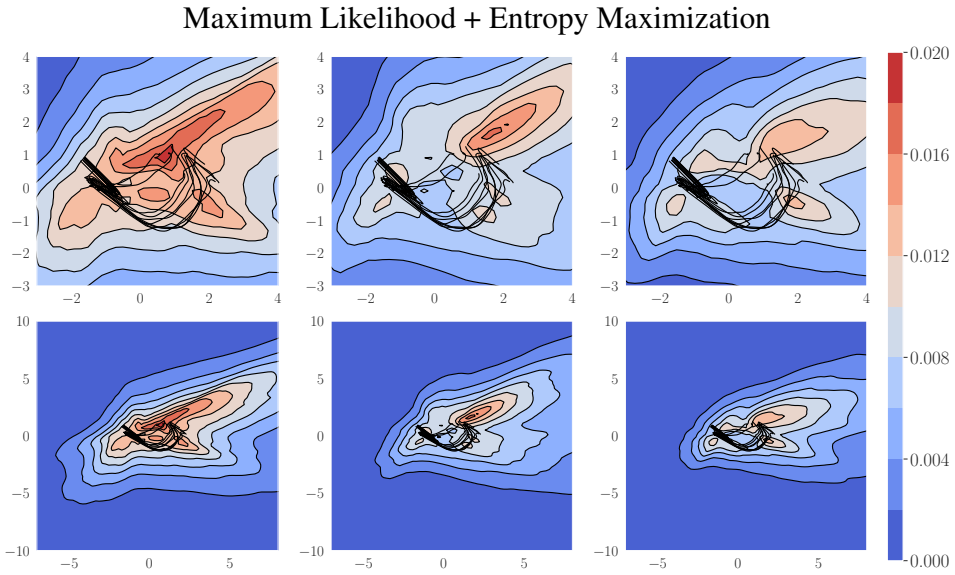


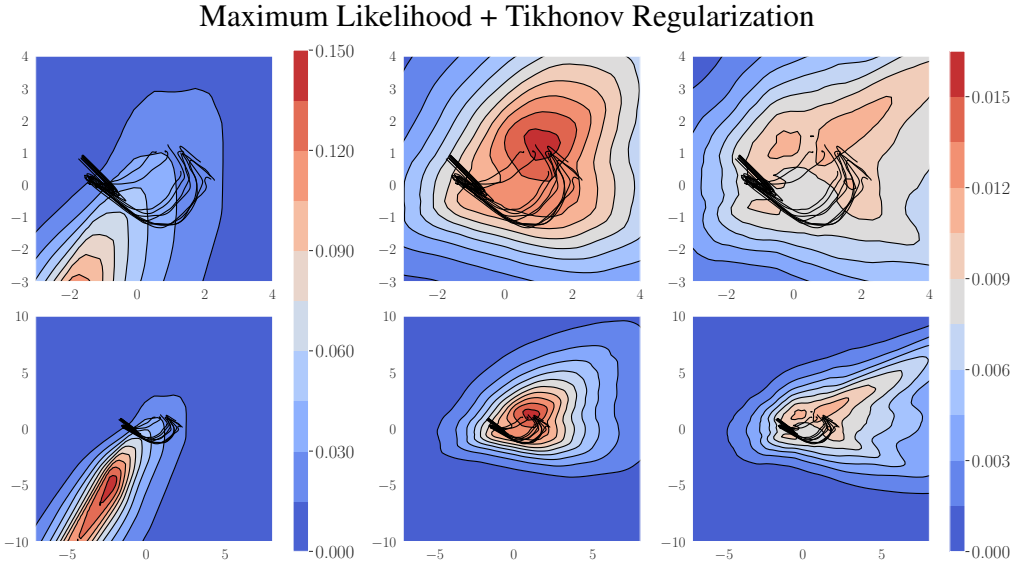
Figure 10: Ablation study using heavier-tailed Laplace behavioral reference policy on MuJoCo locomotion tasks.

B.4 Visualizations of Regularized Maximum Likelihood Parametric Behavioral Policies



(a) $\beta = 10^{-2}$ **(b)** $\beta = 10^{-3}$ **(c)** $\beta = 10^{-4}$

Figure 11: Predictive variances of parametric neural network Gaussian behavioral policies $\pi_\psi(\cdot \,|\, \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{s}), \boldsymbol{\sigma}^2_\psi(\mathbf{s}))$ trained with different entropy regularization coefficients β.



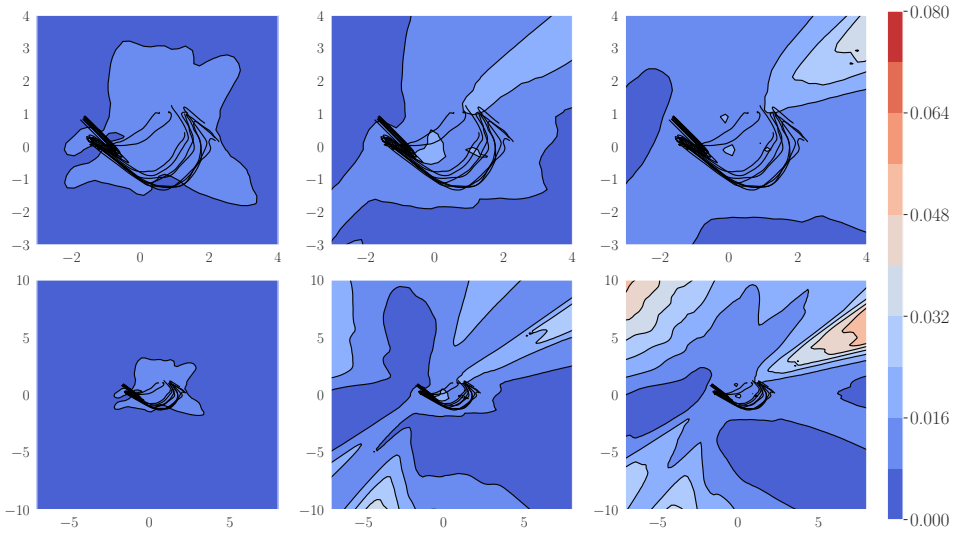
(a) $\lambda = 10^{-2}$ **(b)** $\lambda = 10^{-3}$ **(c)** $\lambda = 10^{-4}$

Figure 12: Predictive variances of parametric neural network Gaussian behavioral policies $\pi_\psi(\cdot \,|\, \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{s}), \boldsymbol{\sigma}^2_\psi(\mathbf{s}))$ trained with different Tikhonov regularization coefficients λ.

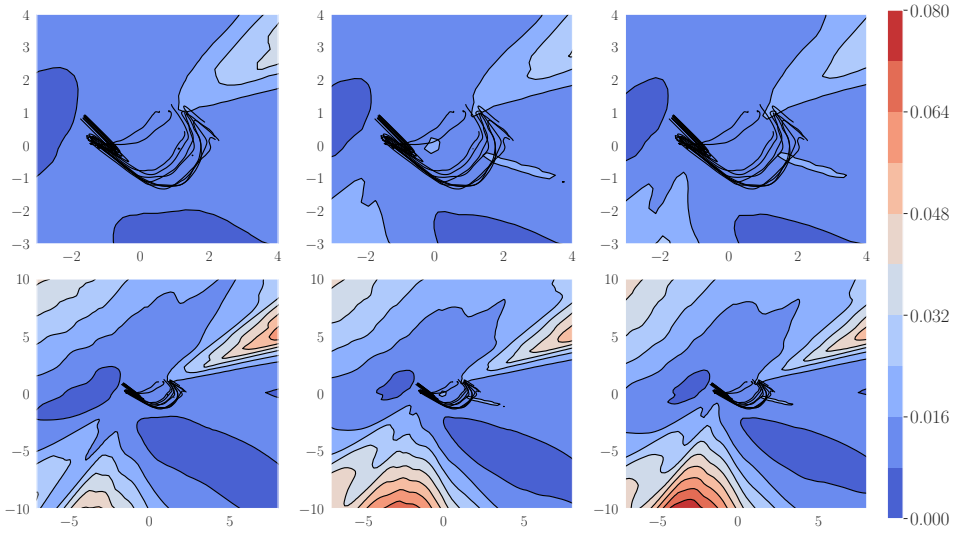
B.5 Visualizations of Ensemble Maximum Likelihood Parametric Behavioral Policies

On the "door-binary-v0" environment, we consider an ensemble of parametric neural network Gaussian policies $\pi_{\psi^{1:K}}(\cdot \,|\, \mathbf{s}) \doteq \mathcal{N}(\boldsymbol{\mu}_{\psi^{1:K}}(\mathbf{s}), \boldsymbol{\sigma}^2_{\psi^{1:K}}(\mathbf{s}))$ with

$$\mu_{\psi^{1:K}}(\mathbf{s}) \doteq \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\mu}_{\psi^k}(\mathbf{s}), \quad \boldsymbol{\sigma}^2_{\psi^{1:K}}(\mathbf{s}) \doteq \frac{1}{K}\sum_{k=1}^{K} \left(\boldsymbol{\sigma}^2_{\psi^k}(\mathbf{s}) + \boldsymbol{\mu}^2_{\psi^k}(\mathbf{s})\right) - \mu^2_{\psi^{1:K}}(\mathbf{s}) \tag{B.15}$$



(a) Single Model **(b)** $K = 2$ **(c)** $K = 4$

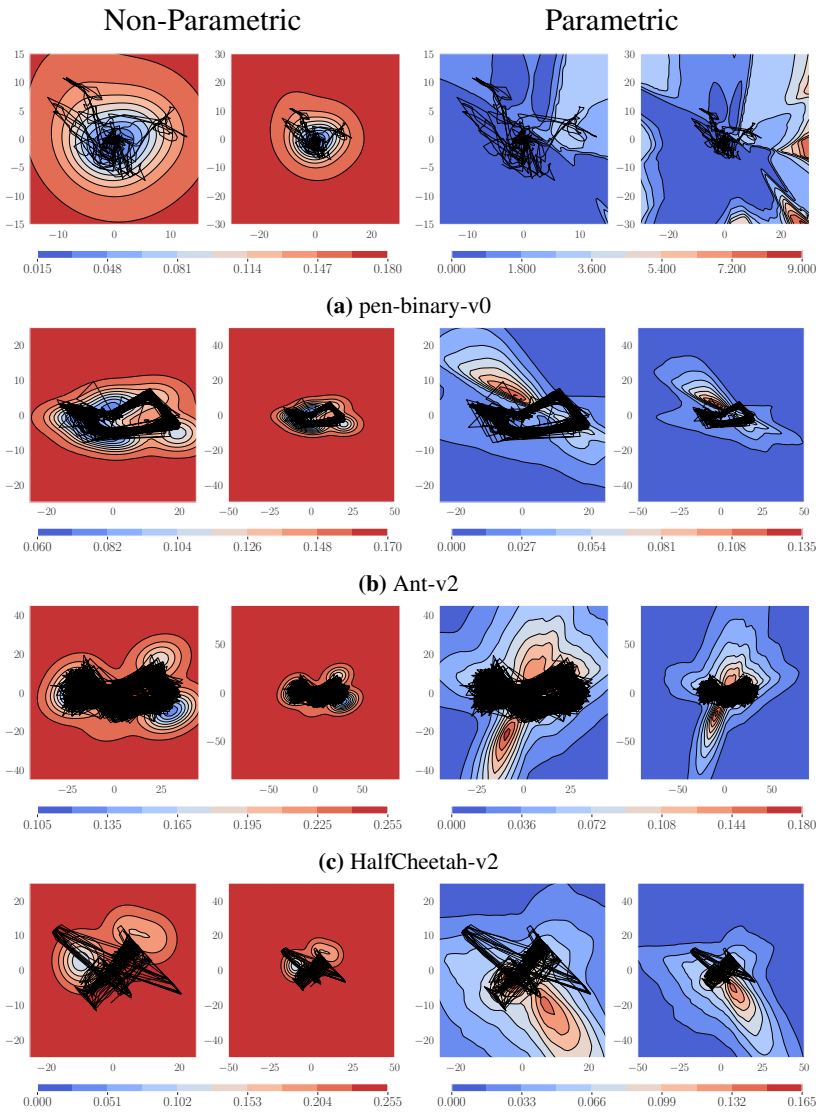


(d) $K = 6$ **(e)** $K = 8$ **(f)** $K = 10$

Figure 13: Predictive variances of ensembles of parametric neural network Gaussian behavioral policies $\pi_{\psi^{1:K}}(\cdot \,|\, \mathbf{s})$ with each neural network in the ensemble trained via MLE. The ensemble policies are marginally better calibrated than parametric neural network policies in that their predictive variance only collapses in some but not all regions away from the expert trajectories.

B.6 Parametric vs. Non-Parametric Predictive Variance Visualizations Across Environments

Figure 14 shows the predictive variances of non-parametric and parametric behavioral policies on low dimensional representations of the environments considered in Figures 4 and 5 (excluding "door-binary-v0", which is shown in Figure 1).



(a) pen-binary-v0

(b) Ant-v2

(c) HalfCheetah-v2

(d) Walker-v2

Figure 14: Predictive variances of non-parametric and parametric behavioral policies on low dimensional representations of the environments considered in Figures 4 and 5 (excluding "door-binary-v0", which is shown in Figure 1). **Left Column**: Non-parametric Gaussian process posterior behavioral policy $\pi_{\mathcal{GP}}(\cdot \,|\, \mathbf{s}, \mathcal{D}_0) = \mathcal{GP}(\boldsymbol{\mu}_0(\mathbf{s}), \boldsymbol{\Sigma}_0(\mathbf{s}, \mathbf{s}'))$. **Right Column**: Parametric neural network Gaussian behavioral policy $\pi_\psi(\cdot \,|\, \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{s}), \boldsymbol{\sigma}^2_\psi(\mathbf{s}))$. Expert trajectories $\mathcal{D}$ used to train the behavioral policies are shown in black. As in Figure 1, the predictive variance of the GP is well-calibrated, whereas the predictive variance of the neural network is not.

B.7 Visual Comparison of Parametric vs. Non-Parametric Behavioral Policy Trajectories

To better understand the significance of the behavioral policy's model class, we sample trajectories from different behavioral policies on the door-opening task in Figure 15. We visualize the mean trajectory and predictive variances of various behavioral policies showing a more sensible mean trajectory and predictive variance from the non-parametric GP policy leading to better regularization compared to a behavioral policy parameterized by a neural network and the implicit uniform prior in SAC, a state-of-the-art RL algorithm. On a randomly sampled unseen goal, we can see in Figure 15b that a neural network policy trained via MLE produces a confident but incorrect trajectory. The starting position is shown in black and the goal position is shown in green. We also visualize a uniform prior, which SAC implicitly regularizes against. Informative priors from offline data can greatly accelerate the online performance of such actor-critic methods.

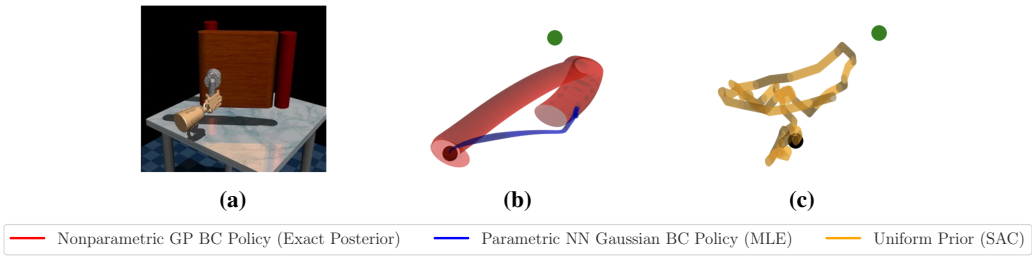


Figure 15: Left: challenging door opening task [35] which standard RL algorithms struggle on. Right and center: 3D plots of sampled mean trajectories and predictive variances from different behavioral policies from expert demonstration π_0, showing a more sensible mean trajectory and predictive variance from the non-parametric GP policy leading to better regularization over both: **(b)** a behavioral policy using a poor model class, and **(c)** the implicit uniform prior in SAC. Starting position shown in **black** and goal position shown in **green**.

Appendix C Further Implementation Details

C.1 Algorithmic Details

Pre-training On the dexterous hand manipulation tasks, before online training, the online policy is pre-trained to minimize the KL divergence to the behavioral reference policy on the offline dataset:

$$J_{\mathcal{GP}}(\phi) \doteq \mathbb{E}_{\mathbf{s}\sim\mathcal{D}_0}\left[\mathbb{D}_{\mathrm{KL}}(\pi_\phi(\cdot\,|\,\mathbf{s}) \parallel \pi_0(\cdot\,|\,\mathbf{s}))\right].$$

C.2 Hyperparameters

Table 2 lists the hyperparameters used for N-PPAC. For other hyperparameter values, we used the default values in the RLkit repository. When multiple values are given, the former refer to MuJoCo continuous control and the latter to dexterous hand manipulation tasks.

Table 2: N-PPAC hyperparameters.

Parameter	Value(s)
optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
reward scale	1
replay buffer size	10^6
number of hidden layers	{2, 4}
number of hidden units per layer	256
number of samples per minibatch	{256, 1024}
activation function	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
number of policy pretraining epochs	400
GP covariance function	{RBF, Matérn}

Table 3 lists the hyperparameters used to train the Gaussian process on the offline data. The hyperparameters are trained by maximizing the log-marginal likelihood. The offline data is provided under the Apache License 2.0.

Algorithm 1 Non-Parametric Prior Actor–Critic

Input: offline dataset $\mathcal{D}_0$, initial parameters θ_1, θ_2, ϕ, GP $\pi_0(\cdot \mid \mathbf{s}) = \mathcal{GP}\big(m(\mathbf{s}), k(\mathbf{s}, \mathbf{s}')\big)$
Condition $\pi_0(\cdot \mid \mathbf{s})$ on $\mathcal{D}_0$ to obtain $\pi_0(\cdot \mid \mathbf{s}, \mathcal{D}_0)$
for each offline batch **do**
  $\phi \leftarrow \phi - \lambda_{\mathcal{GP}} \hat{\nabla}_\phi J_{\mathcal{GP}}(\phi)$ ▷ Minimize KL between online and behavioral reference policy.
end for

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights.
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\cdot \mid \mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Minimize J_Q and J_π using GP $\pi_0(\cdot \mid \mathbf{s}, \mathcal{D}_0)$.
    $\hat{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\hat{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights.
  end for
end for
Output: Optimized parameters θ_1, θ_2, ϕ

Table 3: GP optimization hyperparameters.

Parameter	Value
optimizer	Adam
learning rate	0.1
number of epochs	500

Hyperparameter Sweep for Section 5.4. For the BNN behavioral policy trained via Monte Carlo dropout, a dropout probability of $p = 0.1$ and a weight decay coefficient $1e-6$ were used. These values were found via a hyperparameter search over $\{0.1, 0.2\}$ for p and $\{1e{-}4, 1e{-}5, 1e{-}6, 1e{-}7\}$ for the dropout probability and the weight decay coefficient, respectively.

For the deep ensemble behavioral policy, $M = 15$ ensemble members and a weight decay coefficient of $1e-6$ were used. The weight decay coefficient was found via a hyperparameter search over $\{5, 10, 15, 20\}$ for M and $\{1e-4, 1e-5, 1e-6, 1e-7\}$ for the weight decay coefficient. Each ensemble member was trained on a different 80-20 training–validation split and initialized using different random seeds.

D

Appendices of Chapter 6

A CALIBRATION

A.1 CHOICE OF CALIBRATION METRICS

We consider both the Spearman rank (ρ) correlation and Pearson bivariate (r) correlation. We believe that the former better represents the actual statistical power of the metric compared to the true distributional shift value, as it is robust to outliers and isn't impacted by distributional shape (i.e., skewness, kurtosis). After all, we do not know if some 'true' $|G^{\pi}_{\hat{M}}(s,a)|$ is even linearly correlated with the MSE values that we report, so naïvely comparing based on bivariate correlation may result in incorrect assessment of penalty efficacy. However, we do also include the Pearson bivariate correlation to gain insight into how the penalty distribution shape changes with design choices. For instance, consider two metrics that have identical Spearman coefficients, but vastly different Pearson coefficients–this implies they have significantly different distributional shapes whilst having the same statistical ranking power. The two correlation coefficients have the further advantage that they are unaffected by the scale of the uncertainty penalty, which can vary widely. Furthermore, algorithms such as MOPO and MOReL will often scale the penalty by some coefficient λ and thus the raw unscaled value is hard to interpret.

A.2 THE USE OF MSE AS THE GROUND TRUTH FOR DETERMINISTIC DYNAMICS

Following Yu et al. (2020), it is possible upper bound the expected performance η_M of a policy π in the true MDP M under training in a world model MDP $\hat{M}$ as follows:

$$\eta_M(\pi) \geq \mathop{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\hat{P}}} \left[R(s,a) - \gamma |G^{\pi}_{\hat{M}}(s,a)|\right] \tag{2}$$

where $R(\cdot,\cdot)$ is the reward function, $\rho^{\pi}_{\hat{P}}$ represents transitioning under the world model dynamics $\hat{P}$ and policy π. The quantity $|G^{\pi}_{\hat{M}}(s,a)|$ can be upper-bounded by an integral probability metric (IPM):

$$|G^{\pi}_{\hat{M}}(s,a)| \leq \sup_{f\in\mathcal{F}} \left|\mathbb{E}_{s'\sim\hat{P}(s,a)}[f(s')] - \mathbb{E}_{s'\sim P(s,a)}[f(s')]\right| =: d_{\mathcal{F}}(\hat{P}(s,a), P(s,a)) \tag{3}$$

where $\mathcal{F}$ is some set of functions mapping $\mathcal{S}$ to $\mathbb{R}$, and P is the dynamics under the true MDP M. As noted in Yu et al. (2020), making assumptions over the functional form of $\mathcal{F}$ induces different distance measures. Restricting $\mathcal{F}$ to the set of 1-Lipschitz functions results in an IPM with the following form:

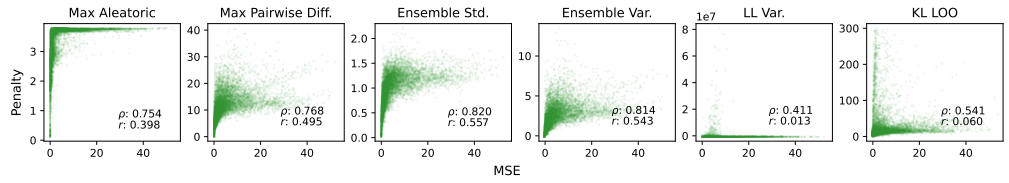
$$|G^{\pi}_{\hat{M}}(s,a)| \leq cW_1(\hat{P}(s,a), P(s,a)) \tag{4}$$

which is the 1-Wasserstein distance, where the constant c is the Lipschitz constant of the value function V^{π}_{M} with respect to a norm $||\cdot||$. Recalling that the environments we evaluate have deterministic dynamics (Todorov et al., 2012), this means the dynamics distributions P and $\hat{P}$ in Eq. 4 are Dirac delta functions. In this case, the 1-Wasserstein distance simply reduces to the 2-norm between some 'true dynamics' $T(s,a)$ and the 'estimated dynamics' $\hat{T}(s,a)$. This justifies the use of MSE between the oracle dynamics (as detailed in Sec. 4.1 and App. D.1) and the world model dynamics as the *ground truth* measure under which we assess calibration.

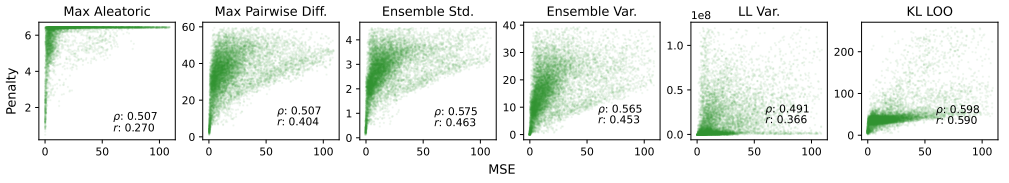
A.3 OFFLINE DATASET TRANSFER CALIBRATION

We present the full calibration scatter plots described in Sec. 4.2. Concretely, we plot penalty values on the y-axis, and ground-truth MSE on the x-axis. First, we present the transfer performance of training sets onto offline datasets. Then, we present the results for all training datasets under the True Model-Based experiment under the adversarial policies.

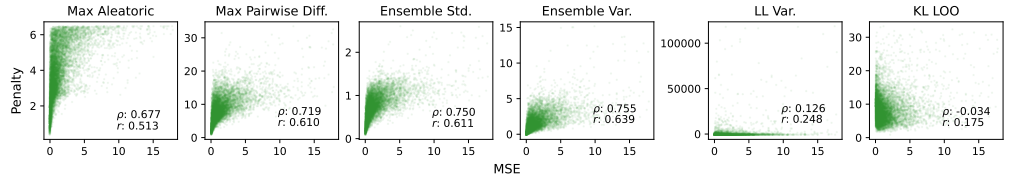
A.3.1 HALFCHEETAH



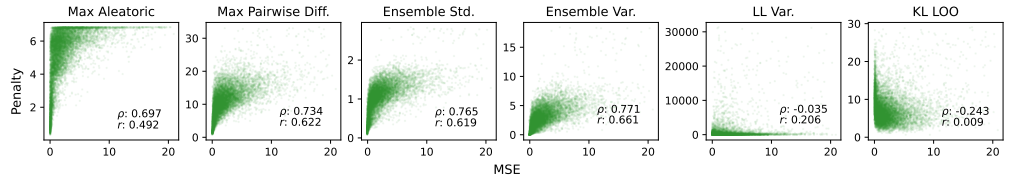
(a) Random transferred to Expert



(b) Medium transferred to Expert



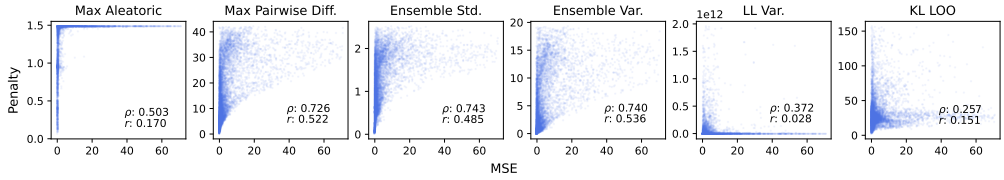
(c) Medium transferred to Random



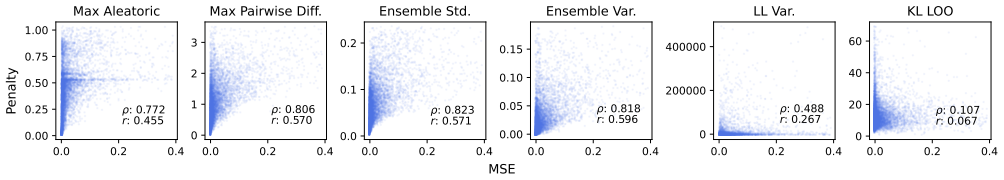
(d) Expert transferred to Random

Figure 5: Scatter Plots showing HalfCheetah D4RL transfer tasks.

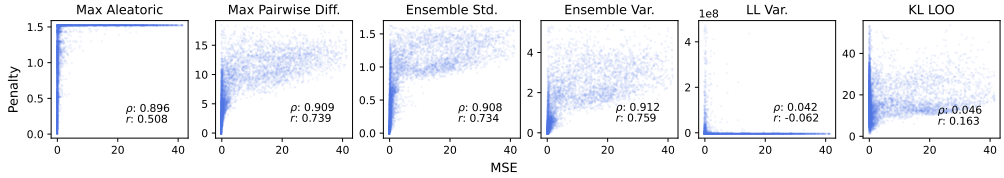
A.3.2 HOPPER



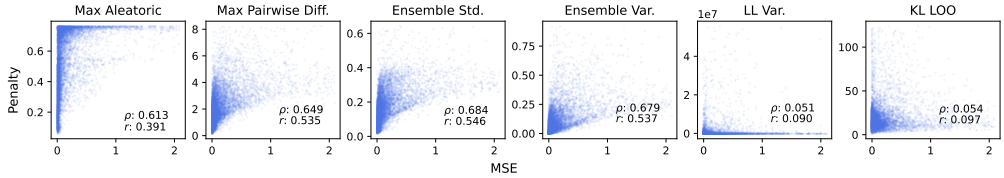
(a) Random transferred to Expert



(b) Medium transferred to Expert



(c) Medium transferred to Random

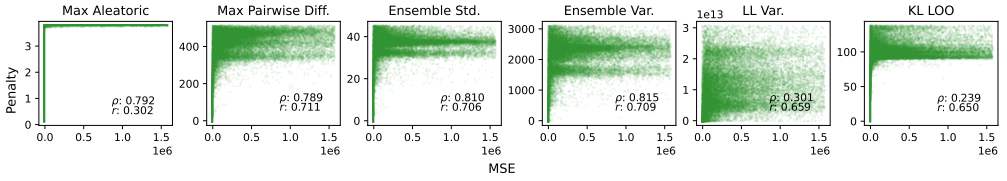


(d) Expert transferred to Random

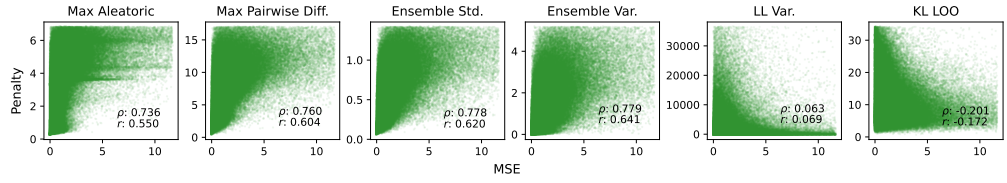
Figure 6: Scatter Plots showing Hopper D4RL transfer tasks.

A.4 TRUE MODEL-BASED ERROR CALIBRATION

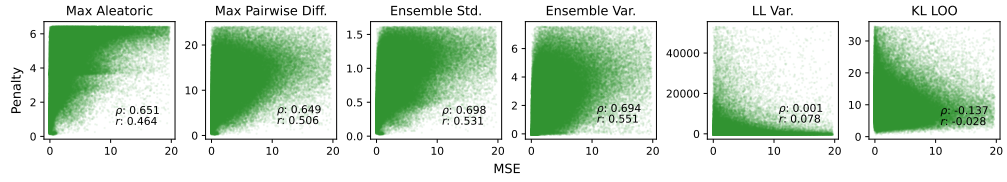
A.4.1 HALFCHEETAH



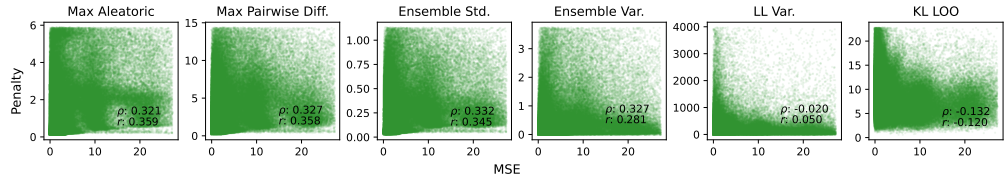
(a) Random



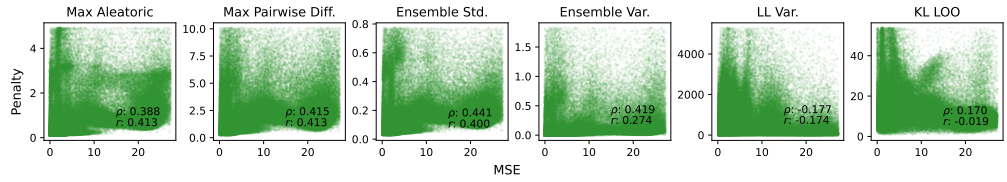
(b) Mixed



(c) Medium



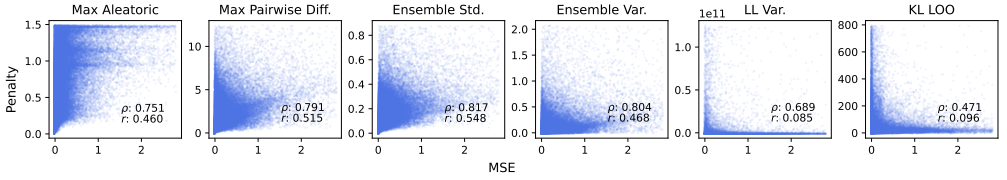
(d) Medium Expert



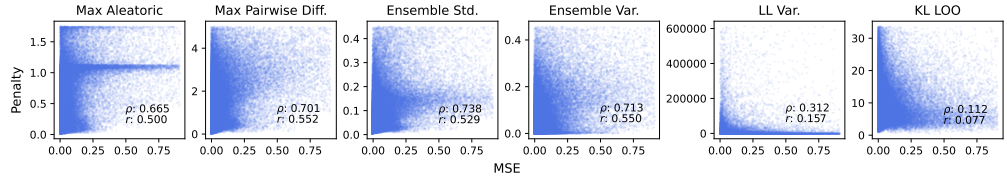
(e) Expert

Figure 7: Scatter Plots showing HalfCheetah D4RL true model-based error calibration.

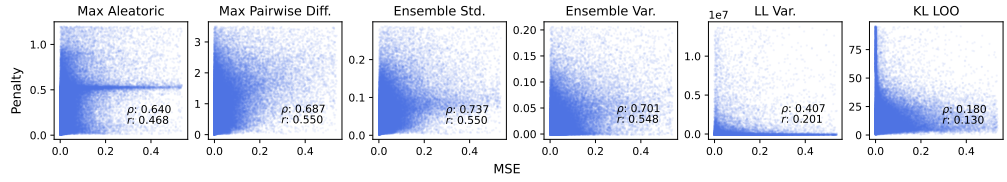
A.4.2 HOPPER



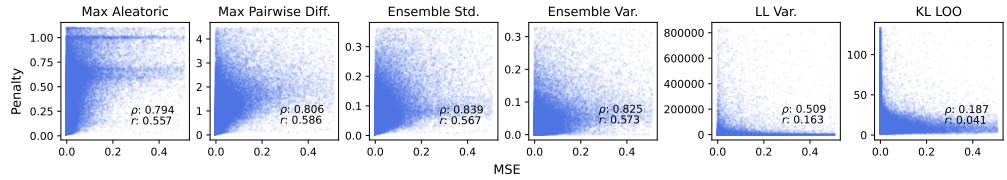
(a) Random



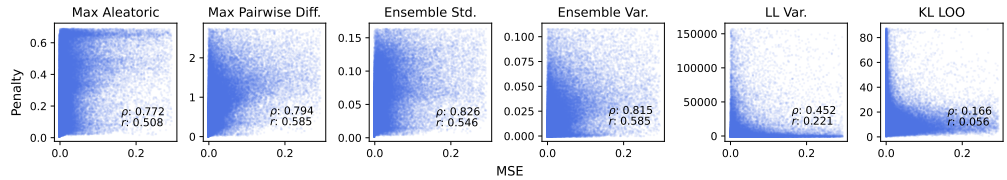
(b) Mixed



(c) Medium



(d) Medium Expert



(e) Expert

Figure 8: Scatter Plots showing Hopper D4RL true model-based error calibration.

A.5 ADDITIONAL LOG PROBABILITY CORRELATION ANALYSIS

Table 5 shows correlation between reward penalties and the negative log-likelihood of the true data under the model in the Transfer experiments.

Table 5: Correlation statistics of penalties against model negative log-likelihood of the true data, averaged over all datasets (i.e., Random through to Expert) showing ± 1 SD over 12 seeds. The best in each column is **bolded**.

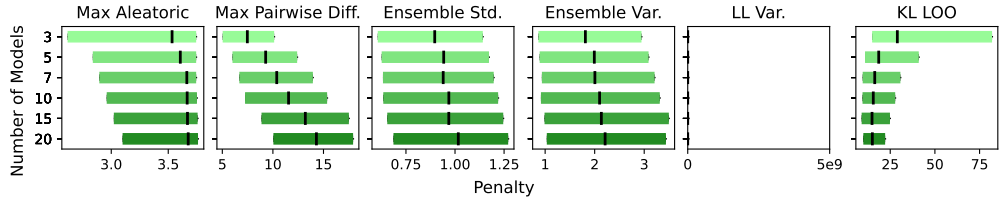
	Transfer			
	HalfCheetah		**Hopper**	
Penalty	ρ	**r**	ρ	**r**
Max Aleatoric	0.87±0.00	0.82±0.01	0.81±0.01	0.57±0.01
Max Pairwise Diff.	0.79±0.01	0.62±0.00	0.79±0.01	0.51±0.00
Ens. Std.	**0.93**±0.00	**0.86**±0.01	**0.89**±0.01	**0.61**±0.01
Ens. Var.	0.90±0.01	0.74±0.01	0.82±0.00	0.59±0.00
LL Var.	0.04±0.07	0.07±0.03	0.25±0.03	0.10±0.01
LOO KL	-0.04±0.06	-0.02±0.04	0.08±0.03	0.05±0.01

B FULL RESULTS INCREASING MODELS

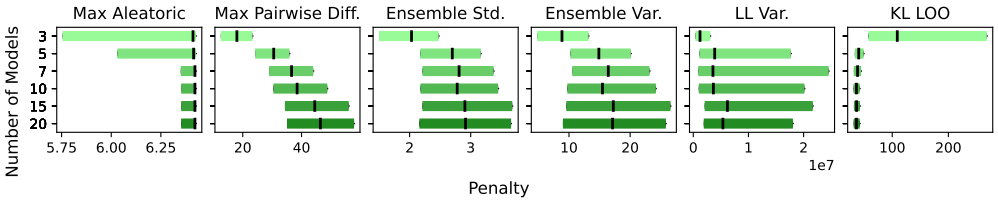
B.1 PENALTY DISTRIBUTION

In this section we provide the full set of results showing the impact of increasing model count on the distribution quantile statistics as introduced in Sec. 5.1. We show inter-quartile range and the median (the latter being denoted by a black vertical line) of each penalty as a function of increasing model number across all training domains and test settings. First, we present the transfer performance of all training sets onto all offline datasets. Then, we present the results for all training datasets under the True Model-Based experiment under the adversarial policies.

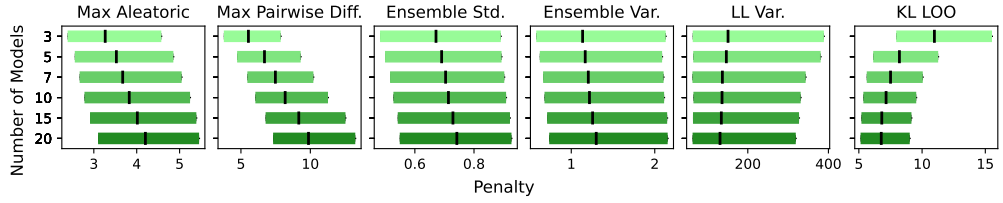
B.1.1 OFFLINE DATASET TRANSFER DISTRIBUTION



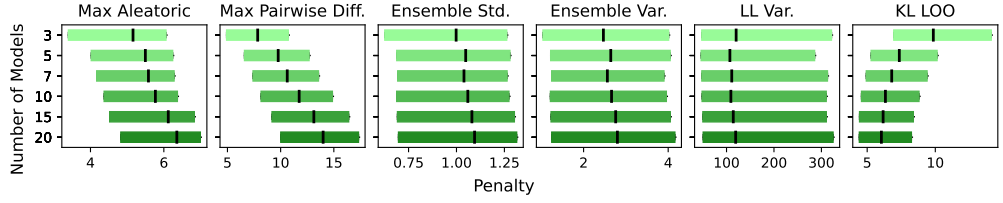
(a) Random transferred to Expert



(b) Medium transferred to Expert

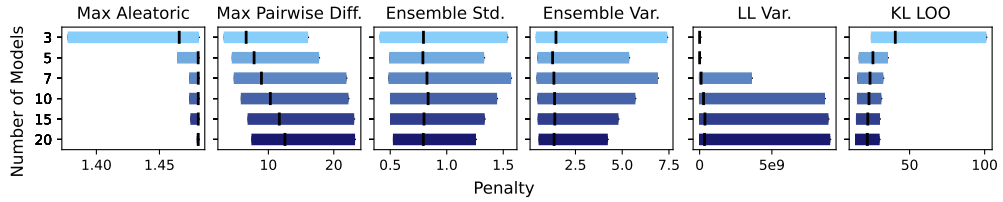


(c) Medium transferred to Random

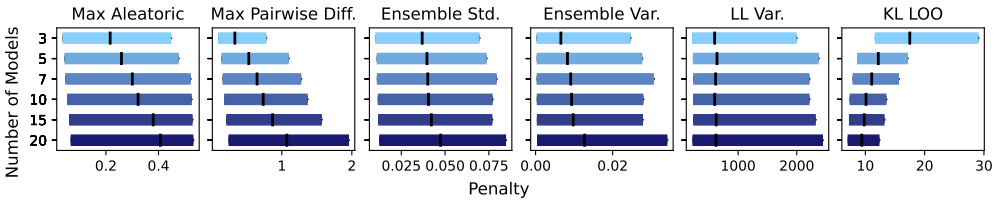


(d) Expert transferred to Random

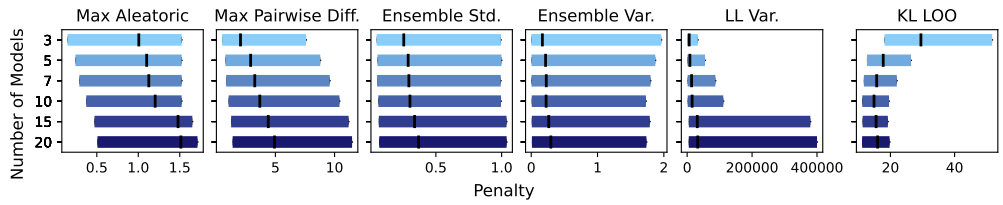
Figure 9: Box Plots showing HalfCheetah D4RL transfer tasks.



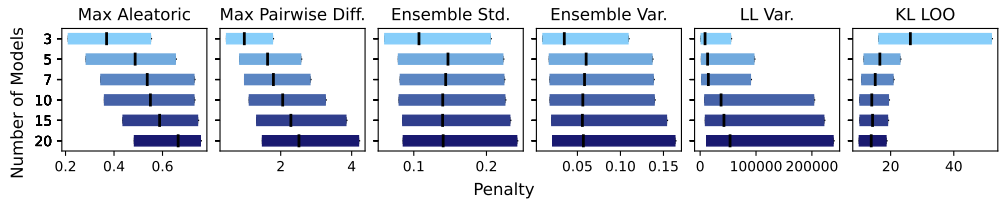
(a) Random transferred to Expert



(b) Medium transferred to Expert



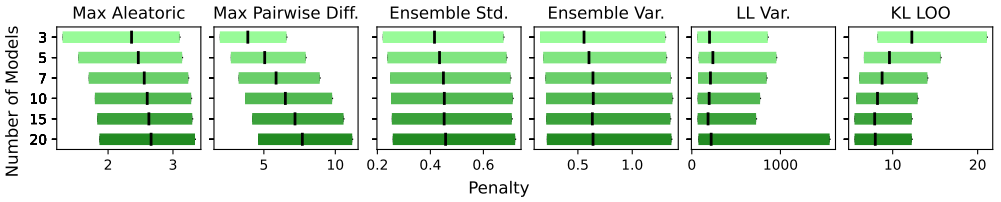
(c) Medium transferred to Random



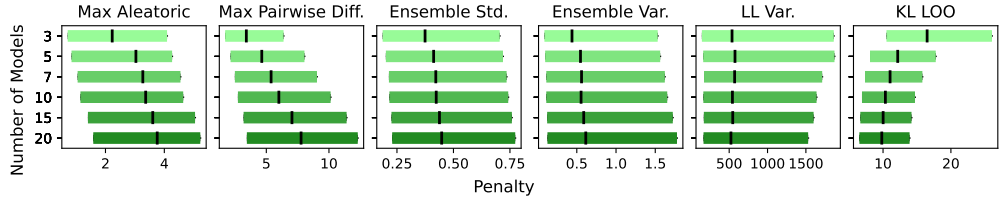
(d) Expert transferred to Random

Figure 10: Box Plots showing Hopper D4RL transfer tasks.

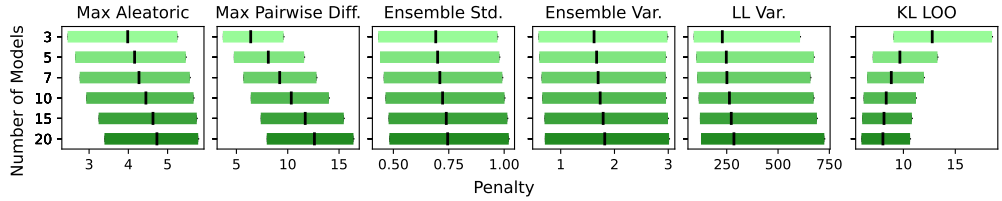
B.1.2 TRUE MODEL-BASED ERROR DISTRIBUTION



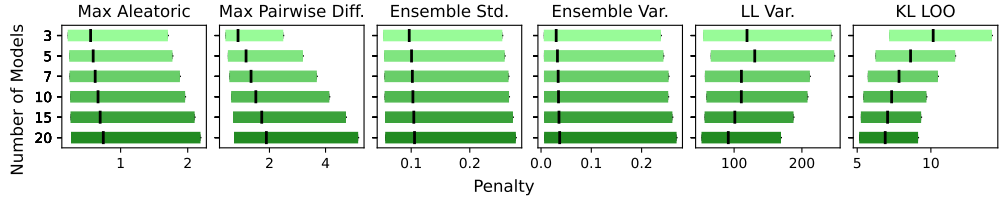
(a) Random



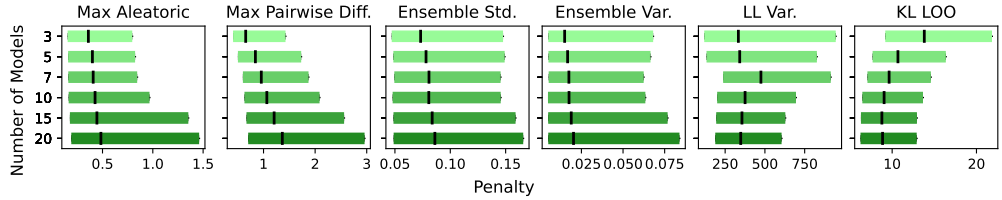
(b) Mixed



(c) Medium

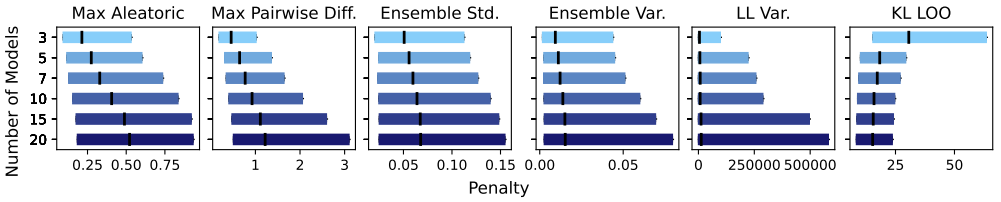


(d) Medium Expert

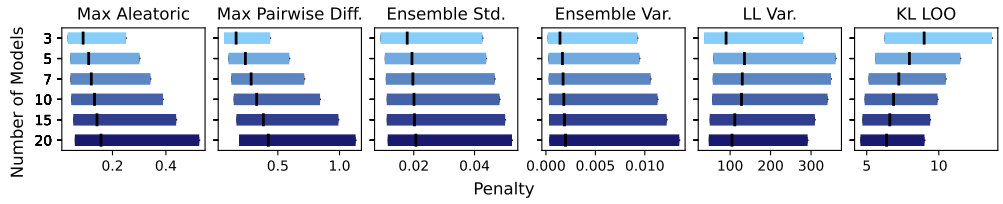


(e) Expert

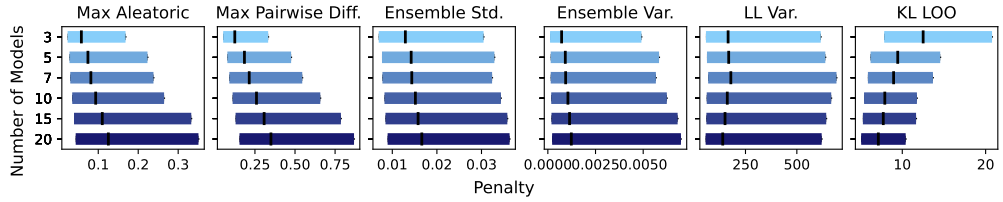
Figure 11: Boxplots showing HalfCheetah D4RL true model-based error penalty distributions.



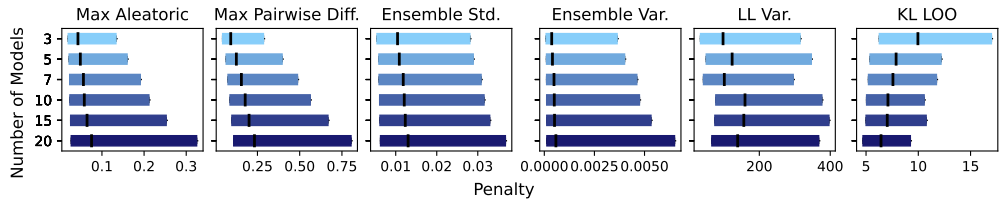
(a) Random



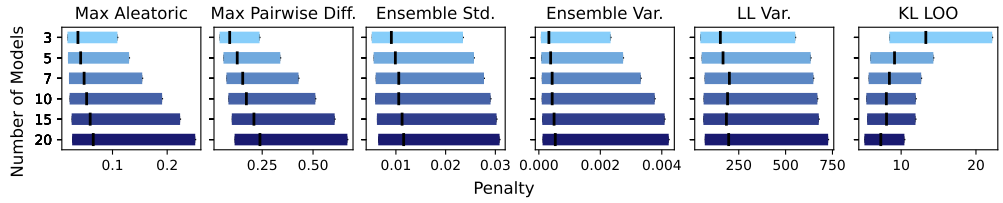
(b) Mixed



(c) Medium



(d) Medium Expert



(e) Expert

Figure 12: Boxplots showing Hopper D4RL true model-based error penalty distributions.

B.2 PENALTY PERFORMANCE

In this section, we provide the full set of results showing the impact of increasing model count on the correlation statistics of each penalty, as described in Sec. 5.1. We show the Spearman and Pearson correlation between penalty and ground truth MSE for all training datasets. First, we present the transfer performance of all training sets onto all offline datasets. Then, we present the results for all training datasets under the True Model-Based experiment under the adversarial policies.

B.2.1 HALFCHEETAH D4RL: TRANSFER

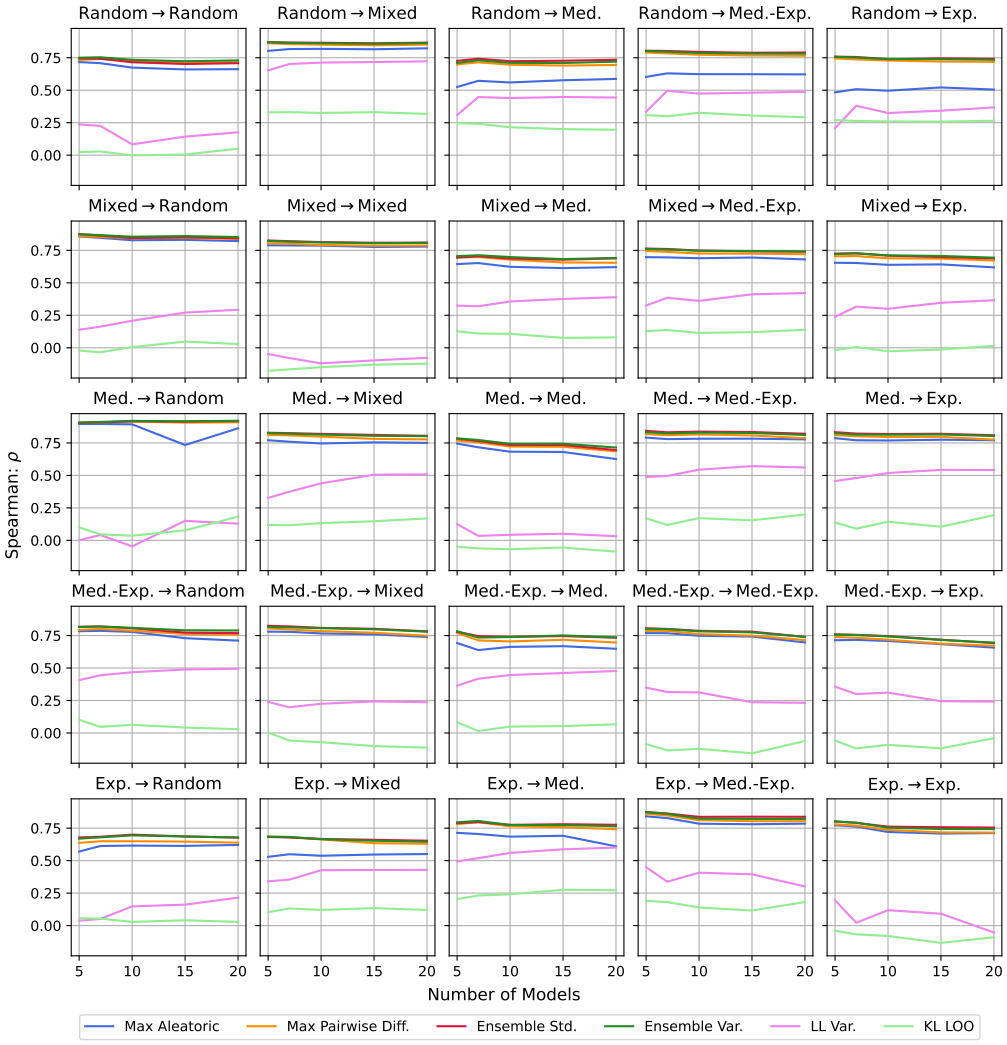


Figure 13: HalfCheetah Spearman Statistics

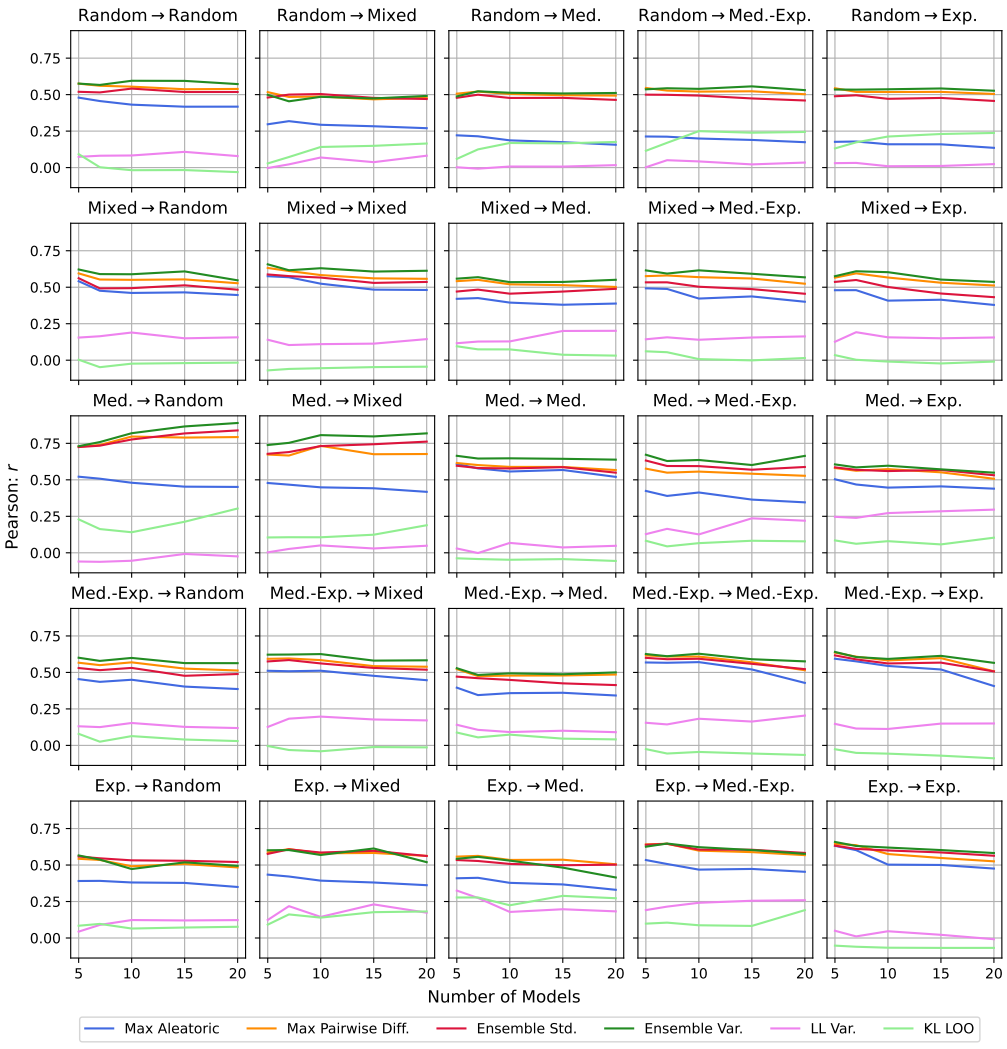


Figure 14: HalfCheetah Pearson Statistics

B.2.2 HOPPER D4RL: TRANSFER

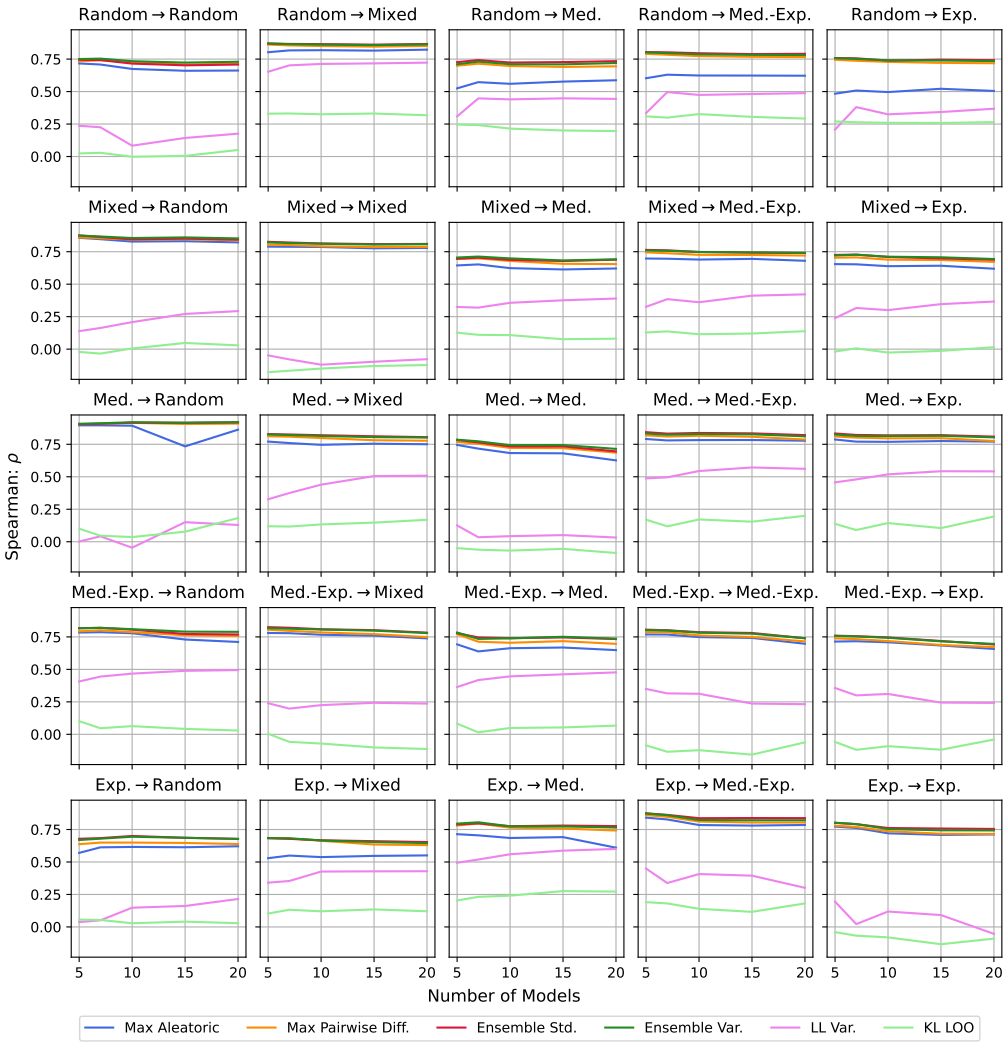


Figure 15: Hopper Spearman Statistics

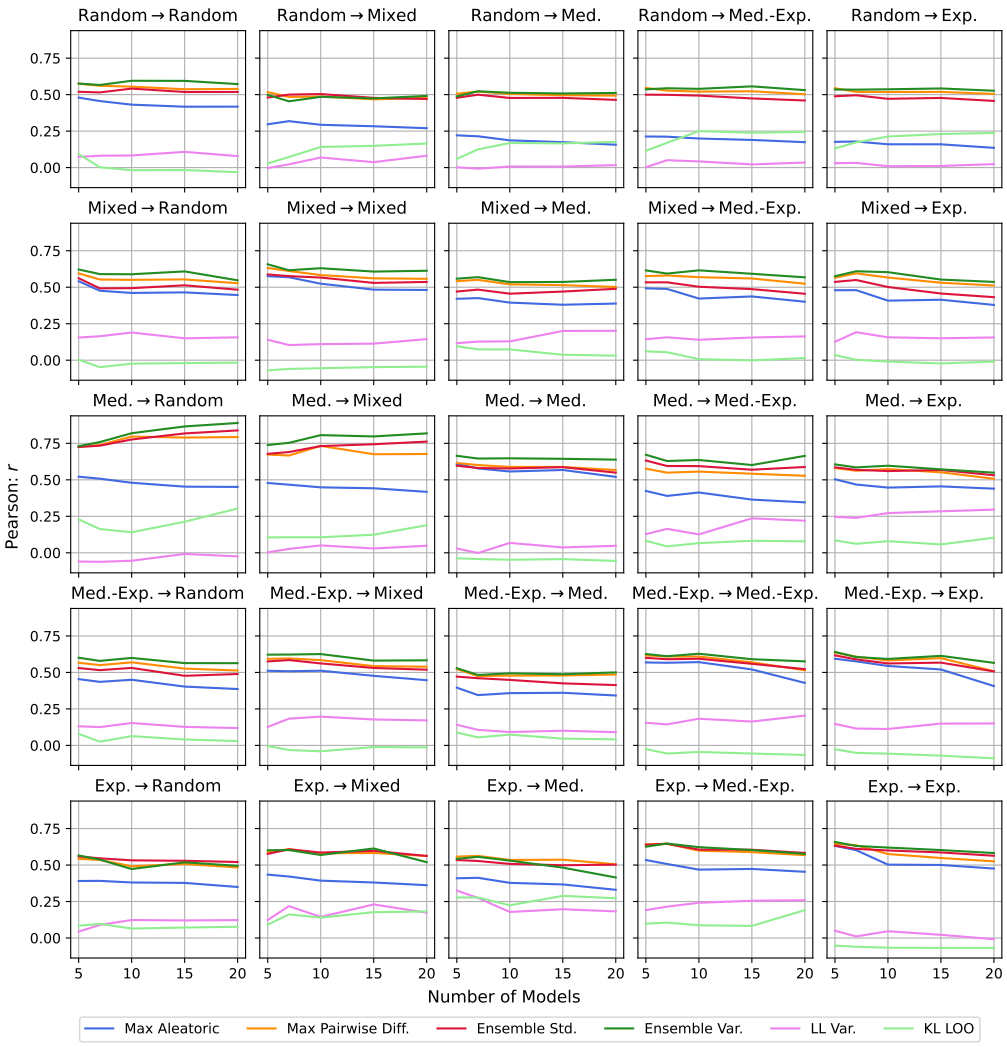


Figure 16: Hopper Pearson Statistics

B.2.3 HALFCHEETAH D4RL: TRUE MODEL-BASED ERROR

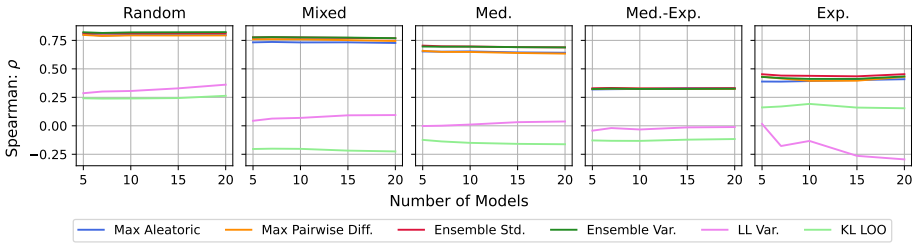


Figure 17: HalfCheetah Spearman Statistics

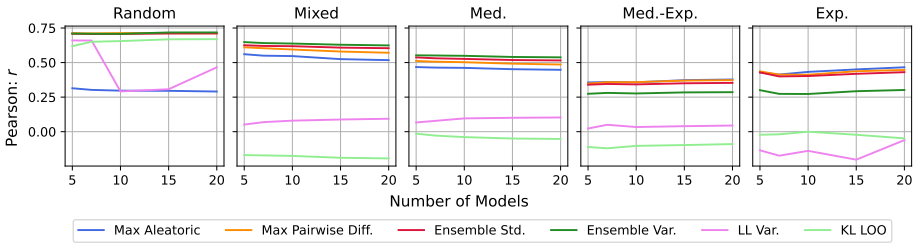


Figure 18: HalfCheetah Pearson Statistics

B.2.4 HOPPER D4RL: TRUE MODEL-BASED ERROR

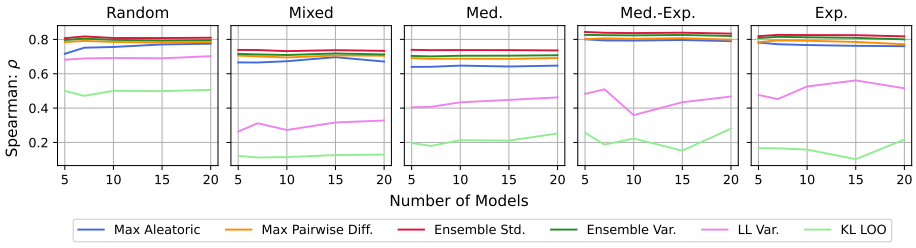


Figure 19: Hopper Spearman Statistics

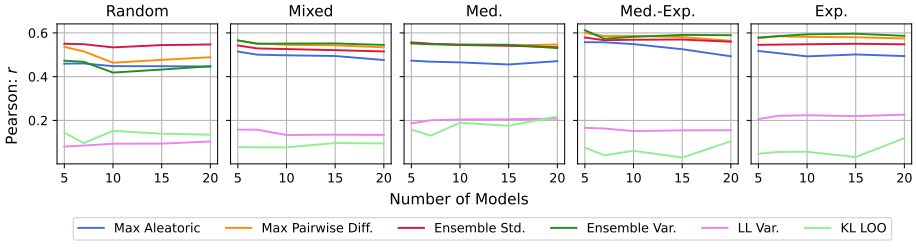


Figure 20: Hopper Pearson Statistics

B.2.5 ALL AGGREGATED

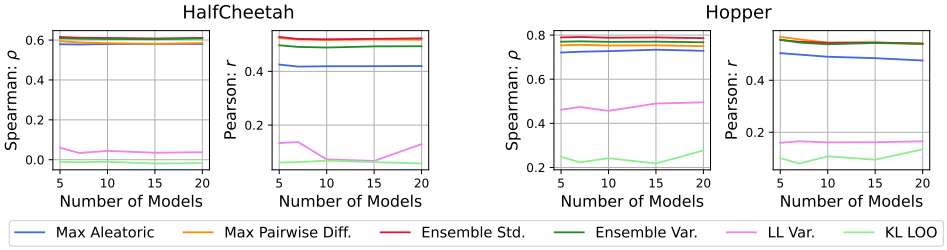


Figure 21: Aggregated True Model-Based correlation statistics over all datasets (i.e., Random through to Expert); **Left:** HalfCheetah; **Right:** Hopper

C SKEWNESS AND KURTOSIS COMPARISONS

C.1 SKEWNESS AND KURTOSIS OVERALL

Here we present the 3rd and 4th order statistics (skew and kurtosis respectively) of each penalty, illustrating that even with identical model counts, the shape statistics between penalties are vastly different.

Table 6: Skew (γ_1) and Kurtosis (γ_2) statistics of all experiments averaged over all datasets (i.e., Random through to Expert) using the MOPO Default of 7 models.

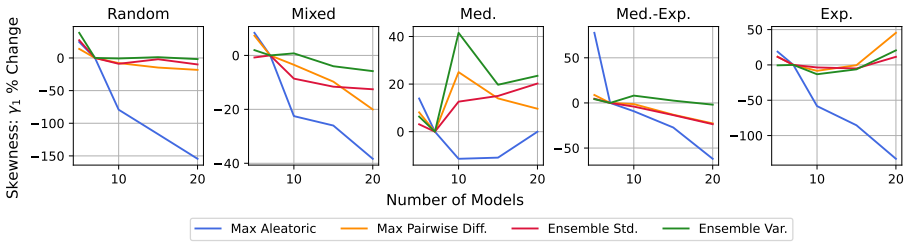
	Transfer				**True Model-Based**			
	HalfCheetah		**Hopper**		**HalfCheetah**		**Hopper**	
Penalty	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
Max Aleatoric	-0.010	0.580	0.689	1.377	0.671	0.920	1.873	2.864
Max Pairwise Diff.	0.919	0.957	1.967	4.578	1.661	3.081	2.571	7.465
Ensemble Std.	0.794	0.806	2.136	6.560	1.656	3.178	2.739	9.061
Ensemble Var.	1.823	4.830	3.436	15.983	2.612	8.800	4.517	25.380
LL Var.	6.893	114.843	10.920	180.716	5.100	37.865	14.415	251.705
LOO KL	1.778	5.729	3.729	29.606	1.840	4.600	4.008	28.089

C.2 SKEW AND KURTOSIS SCALING WITH MODEL COUNT

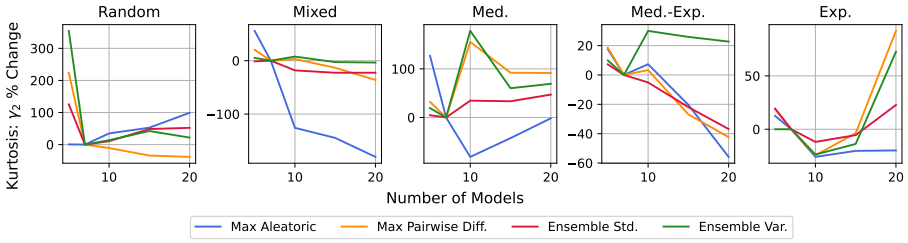
We omit LL Var. and LOO KL due to the fact that their changes were so significant as to obfuscate the changes of the more performant penalties.

We choose 7 models, as in Table 6, to act as our 'baseline' (following the default MOPO setting), and we measure the change in the skew and kurtosis relative to this, hence 7 models always has a 0% change in our graphs. For brevity, in the transfer experiments, we average over all 'transferred to' environments, e.g., Random, Medium, etc.; the graph title refers to the data that the model was trained on.

Again, we observe the environment *and* setting dependency of these metrics, sometimes having increasing skewness and kurtosis with model count, and other times decreasing. This further justifies using a ranking metric to compare penalties, as the overall penalty shape can vary hugely and unpredictably w.r.t. co-dependent hyperparameters. We do observe however in the True Model-Based experiments that ensemble standard deviation appears to be most robust to scaling with models. We also observe that the Max Aleatoric penalty can change shape significantly w.r.t. model count, and no penalties are fully immune to this. This further advocates the use of shape meta-parameters to control for changing distribution properties when adjusting the number of models as a hyperparameter, as well as selecting penalties that are relatively invariant to model count to make tuning easier.

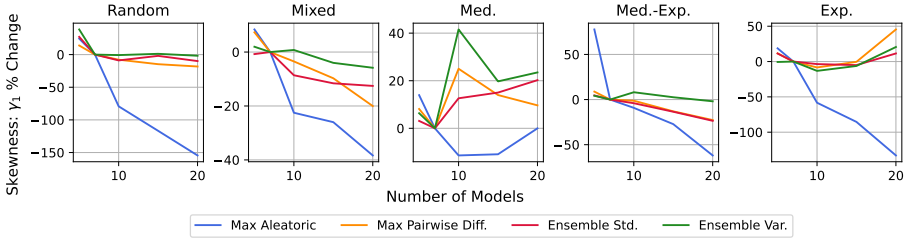


(a) Skewness Scaling

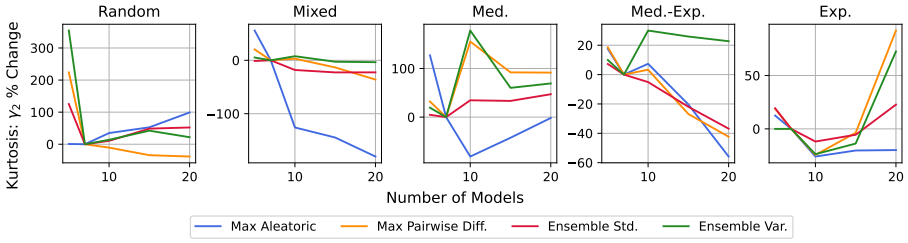


(b) Kurtosis Scaling

Figure 22: HalfCheetah Transfer.

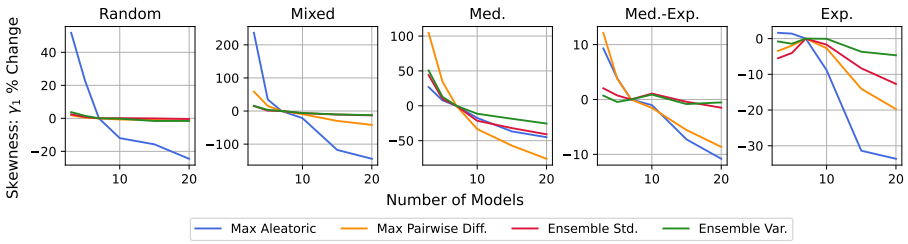


(a) Skewness Scaling

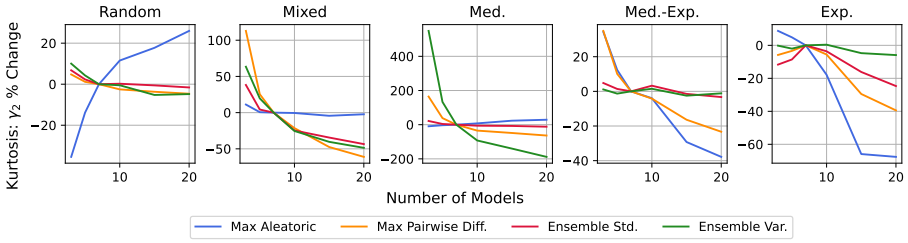


(b) Kurtosis Scaling

Figure 23: Hopper Transfer.

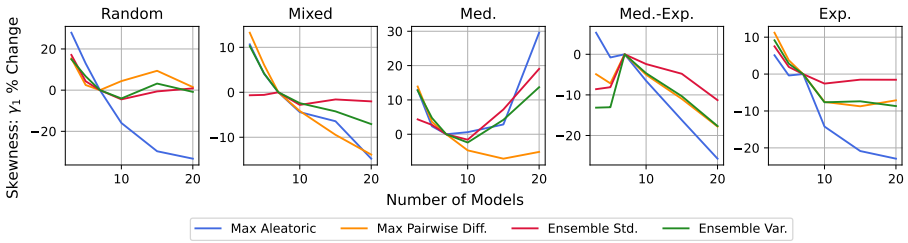


(a) Skewness Scaling

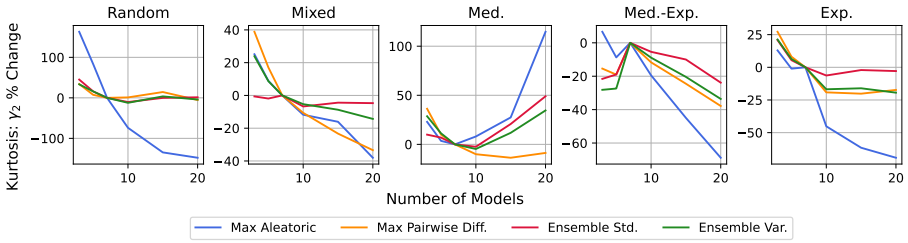


(b) Kurtosis Scaling

Figure 24: HalfCheetah True Model-Based.



(a) Skewness Scaling



(b) Kurtosis Scaling

Figure 25: Hopper True Model-Based.

D FURTHER DETAILS ON TRUE MODEL-BASED EXPERIMENTS

D.1 METHODOLOGICAL DETAILS

We leverage the MuJoCo (Todorov et al., 2012) simulator to provide us with ground truth dynamics that we can use to compare against our model predictions and penalties. This is done by providing the state and action inputs given to the model to the simulator through the `set_state` method in the simulator API. It must be noted that this method also requires an addition 'displacement' value which is not modelled by the world models (nor is it provided in the D4RL data), however we found in practice this did not affect the dynamics predicted by the simulator, and simply setting this to 0 was sufficient to generate ground truth predictions.

This makes it possible to provide the simulator the hallucinated model states, and provide a true proxy to the *dynamics* discrepancy. We note that since the states are 'hallucinated' by the model, it might be the case that they may not be admissible under the true environment, but in reality the simulator was able to process almost any combination of state and action, barring settings that featured anomalously large magnitudes. To handle such cases, we found it necessary to clip the model states to the range $[-10, 10]$.

In order to assess the permissibility of states, as well as measure the accuracy of the penalties as OOD input detectors, we provide an alternative distance measure based on the distance away from the training set. We use this measure for our analysis in Section 5.3, and is calculated as the distance from the offline training dataset, which we define to be the 2-norm between a given state-action tuple and its nearest point in the offline data, a similar metric to those used in recent works on imitation learning (Dadashi et al., 2021). We describe this quantity henceforth as 'Distribution Error'.

D.2 ON THE NATURE OF OOD DATA ALONG HALLUCINATED TRAJECTORIES

Here we discuss the nature of OOD data along a single hallucinated trajectory (in the model) in offline MBRL, analyzing the inductive bias that some 'error' increases with increasing rollout length in the model. We find that there is merit to this assumption, and show this in Fig. 26 for all HalfCheetah and Hopper environments in D4RL. Here, we plot the median error at each time-step across $30,000$ aggregated trajectories in the model. Note that for all plots, we re-normalize all penalties by subtracting their mean and dividing by their standard deviation to facilitate comparison; this normalization was also applied in the analysis performed in Fig. 1a. Concretely, each time step corresponds to the normalized median value of $30,000$ data-points.

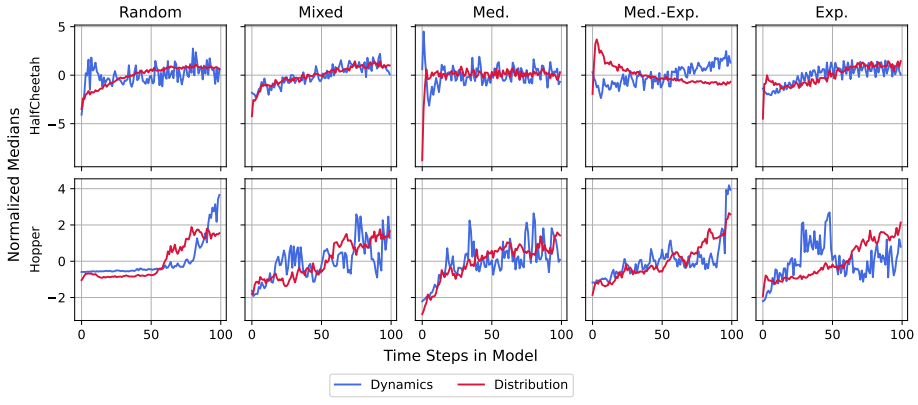


Figure 26: Median True Model-Based Errors as a function of rollout timestep

We observe indeed that both median dynamics and distribution errors increase with increasing time step in the model. The only real exception is HalfCheetah Medium-Expert, which we believe to be due to our trained policy not being able to successfully exploit this environment.

The above analysis captures *overall trends* in the error over a large number of trajectories. However, the way errors manifest during an *individual rollout* is not so straightforward. To illustrate this,

observe Fig. 27, where we plot a random subset of 5 individual rollouts from the Hopper Medium-Expert data we generated.

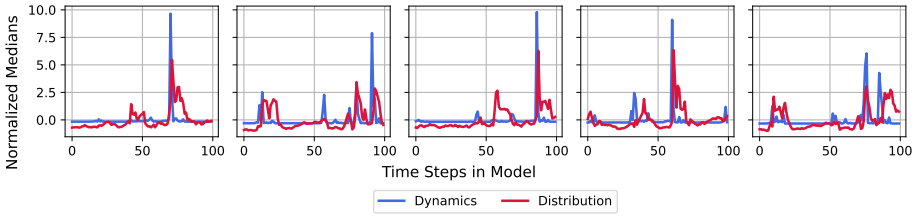


Figure 27: Several Individual Ground Truth Rollouts in Hopper Medium-Expert

We observe that errors along any single trajectory tend to manifest as 'spikes', and that it is entirely possible to recover from these, returning to either admissible dynamics, or parts of the state-action space that have been seen in the data. This speaks to the nature of how we ought to penalize policies for accessing regions of inaccuracy/uncertainty, and may justify a hybrid MOPO/MOReL approach, whereby we penalize individual transitions along a trajectory, but do not stop rollouts early. Indeed, this is similar to the approach taken in M2AC (non-stop), albeit they choose to 'mask' uncertain transitions, not penalize them. We leave the design of such an algorithm to future work.

Finally, we address the issue of comparing OOD dynamics and inputs. As already observed in Fig. 27, these two errors are not necessarily always the same, and oftentimes it is possible that one quantity is large, whilst the other is small. We revisit Fig. 1a to explore this, now also plotting the Distribution Error in Fig. 28.

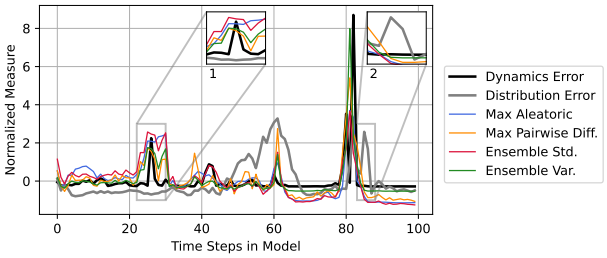


Figure 28: Comparing OOD dynamics and inputs on a Hopper Medium-Expert trajectory

We first speak to the inset annotated '1'. Here we observe that the transitions generated in fact closely resemble the data that our model was trained on, however the predicted dynamics are incorrect, and cause an aforementioned 'spike'. This is the opposite of what is observed in the inset annotated '2'; where we actually predict accurate dynamics, however the resultant state-action tuples do not closely resemble the data that our model was trained on. We generally observe that regions of high Distribution Error tend to be preceded by 'spikes' pertaining to high Dynamics Error, and this present an exciting avenue for future work understanding how these quantities are related.

D.3 On the diversity of exploitative policies

It is possible that training policies purely to exploit the world models may result in generating state-action tuples that are low in diversity, as the policy could discover "pockets" in the model that provide consistently high return. To prevent this, we train multiple policies inside the model from different seeds, with the aim of inducing different modes of exploitation. To validate this induces diverse trajectories in the world models, we visualize the state-action manifold using a t-SNE projection (van der Maaten & Hinton, 2008) of: 1) the D4RL Hopper Mixed-Replay (which contains diverse samples); 2) the imagined rollouts inside the model from the exploitative policies in Fig. 29. The policies were trained to exploit a model that itself was trained on the Hopper Mixed-Replay data.

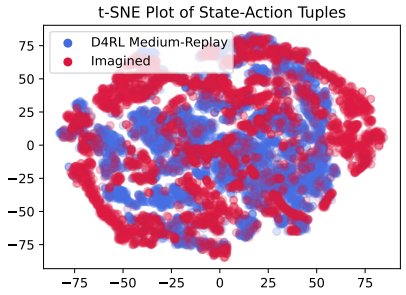


Figure 29: t-SNE projection of 10,000 Hopper D4RL Medium-Replay and 10,000 exploitative Imagined policy state-action tuples

We observe that the induced policies inside the model displays some overlap with the D4RL data, but also resides in parts of the manifold where there is little coverage from the D4RL data, likely representing regions of exploitation. Importantly, the exploitative WM trajectories display strong state-action tuple diversity, comparable to that of the offline data it was trained on.

E USING METRICS AS OOD EVENT DETECTORS

E.1 MEASURING STATISTICS

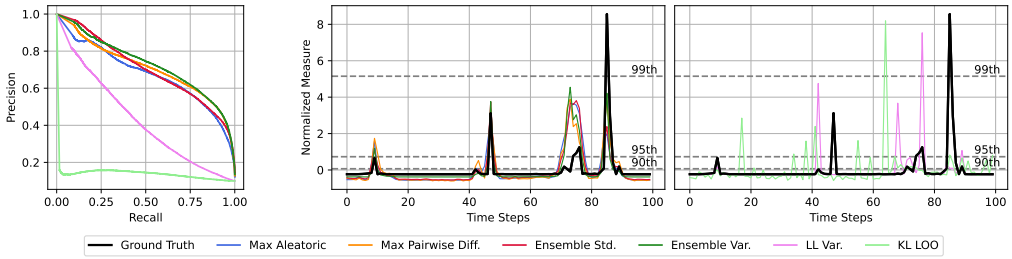


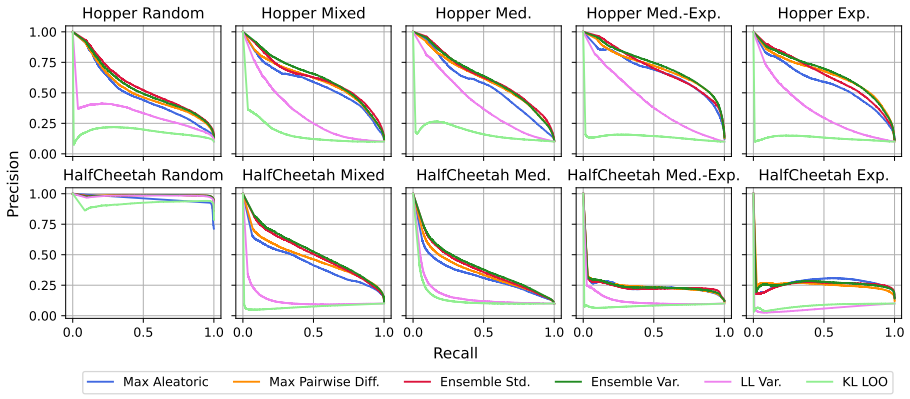
Figure 30: Hopper Medium-Expert True Model-Based Experiments; **Left:** Precision v.s. Recall against Ground Truth; **Middle:** Higher Performing Penalties v.s. Ground Truth MSE in Imagined Rollout; **Right:** Lower Performing Penalties v.s. Ground Truth MSE in Imagined Rollout

As noted previously, different penalties have varying scales and distribution profiles, so we need a way of standardizing the method of assessment. Using our observation that errors manifest as 'spikes' during a rollout, we propose treating each penalty as a classifier. Concretely, our test set consists of the ground truth data labeled by whether or not they exceed a certain percentile at a particular time step. Each penalty may be then be treated as a 'classifier' by normalizing its range to lie in $[0, 1]$. We can then use standard classification quality measures, such as AUC, to determine the effectiveness of these penalties at capturing these spikes, whilst sidestepping the issue of the different distributional profiles identified previously.

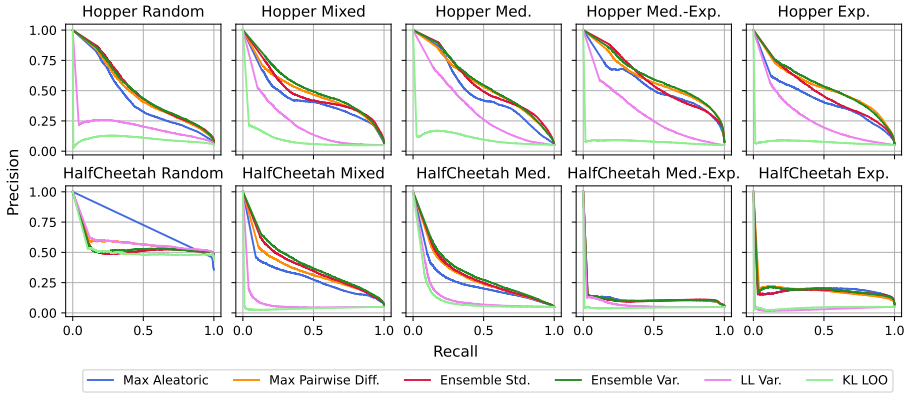
Fig. 30 shows how our proposed method may be used to compare the effectiveness of each metric at capturing OOD events. In the figure, we plot a single rollout in the model, and the resultant ground truth MSE between the predicted next state and the true next state in black. We then superimpose the 90th, 95th and 99th percentile MSEs across the entire imagined trajectories onto the figure in gray dashed lines. To construct our OOD labels, we label any point below the percentile line as being 'False', and any point above that line as being 'True'. Finally, we normalize the uncertainty metrics as previously described into values in the range $[0, 1]$, allowing us to construct precision-recall graphs and calculate classifier statistics.

E.2 PRECISION RECALL CURVES

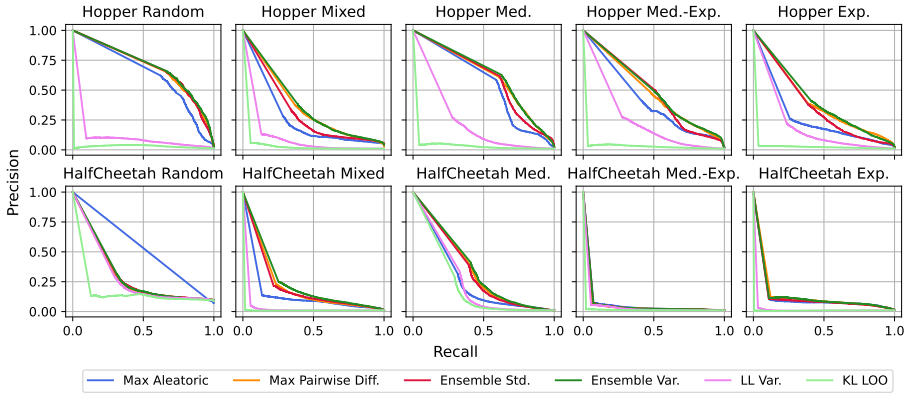
In this section we present the Precision-Recall curves described in App. E.1.



(a) 90th Percentile



(b) 95th Percentile



(c) 99th Percentile

Figure 31: Precision Recall curves on ground truth data.

F IMPLEMENTATION DETAILS

This paper extensively discusses key *hyperparameters* specific to current offline MBRL algorithms. However, there are significant *code-level* implementation details which are often critical for strong performance and make it hard to disambiguate between algorithmic and implementation improvements. Worryingly, many of these details are not mentioned in their respective papers, or are different between the authors' code and paper. We detail clear examples of this below. We believe further investigation of these code-level implementation details represents important future work, as has already been done for policy gradients (Engstrom et al., 2020; Andrychowicz et al., 2021). Indeed – it is unclear if the improvement of MOReL over MOPO is due to its different P-MDP formulation, or if it is successful *in spite of* this formulation, due to a superior policy optimization strategy or dynamics model design. We believe that this paper takes a significant first step in tackling this issue by directly comparing a number of key design choices, and understanding their individual impact. Now we summarize key differences between the paper and code for the MOPO and MOReL algorithms which we compare against that are crucial to achieve the same reported performance.

In MOPO,

- Each layer in the model neural network has a different level of weight-decay
- The authors' code uses different objectives for training (log-likelihood) and validation (MSE).
- The authors use elites, but only for next state prediction (as discussed previously).

In MOReL,

- There is a difference in the authors' code about how the penalty threshold is calculated and tuned, and isn't provided as a hyperparameter in the appendix.
- The absorbing HALT state does not appear in the authors' code.
- The negative halt penalty appears significantly different between code and paper.
- There is a minimum trajectory steps parameter (hard-coded to 4) not mentioned in the paper.
- The reward function appears to be hard-coded in the authors' implementation, not learned as stated in the paper.
- The policy architecture is different in the authors' code (64,64 hidden layers) and the paper (32,32 hidden layers)
- It is not clear when the optional behavior cloning initialization step is applied.

G HYPERPARAMETERS AND EXPERIMENT DETAILS

The D4RL (Fu et al., 2021a) codebase and datasets used for the empirical evaluation is available under the CC BY 4.0 Licence. As stated in the main text, we choose to use 'v0' experiments as these are more challenging for Hopper due to having low return trajectories (Kostrikov et al., 2021), and we clearly state when other benchmarks use the 'v2' experiments, which have offline trajectories with higher returns on Hopper.

The remaining hyperparameters for the MOPO algorithm that we do not vary by Bayesian Optimization were taken from the original MOPO paper (Yu et al., 2020), apart from we fix the number of policy epochs/iterations to 1,000 for all experiments. This means our implementation uses the same probabilistic dynamics models (with unchanged hyperparameters) and policy optimizer (SAC, Haarnoja et al. (2018)) as MOPO, differing from MOReL, which uses Natural Policy Gradient (Kakade, 2002).

The hyperparameters used for the BO algorithm, CASMOPOLITAN, are listed in Table 7. We use the batch-mode of CASMOPOLITAN, where multiple hyperparameters settings are proposed and evaluated concurrently.

Each BO iteration is run for 300 epochs on a single seed, and the full optimization over an offline dataset took ~200 hours on a NVIDIA GeForce GTX 1080 Ti GPU taken up predominantly by MOPO training.

Table 7: CASMOPOLITAN Hyperparameters

Parameter	Value
Number of parallel trials	4
Number of random initializing points	20
ARD	False
Acquisition Function	Thompson Sampling
Global BO	True
Kernel	CoCaBo Kernel (Ru et al., 2020)

Unless specified otherwise, plots and reported statistics are completed with 7 models in the ensemble, as this is the number chosen in the original MOPO paper used with the Max Aleatoric penalty.

H RLIABLE FRAMEWORK FOR PERFORMANCE EVALUATION

Throughout this work, we choose to adopt the `rliable` framework introduced in Agarwal et al. (2021) to evaluate the performance of our approaches. `rliable` advocates for computing aggregate performance statistics and probability of improvement across many tasks in a benchmark suite; indeed we take this approach when reporting the values in the analysis performed in Sections 4 and 5. This is important when the number of tasks become large, and also prevents outliers from dominating mean statistics. Furthermore, this allows us to make clear statements about improvements given the relatively low number of seeds that are used in deep RL; indeed, Agarwal et al. (2021) show that even using high seed counts does not ameliorate the variance issues experienced when training such algorithms. For normalization, we use the standard D4RL return scaling.

I EVALUATION USING AUTOMATIC CONSTRAINT TUNING

We show the full tabulated results from Sec. 6 with statistical significance using the `rliable` framework in Table 8, using the 'Probability of Improvement' metric in Agarwal et al. (2021).

To evaluate our claims in Sections 4 and 5 without needing to laboriously tune the penalty weight λ per environment, we employ an automatic penalty tuning scheme, analogous to the automatic entropy tuning used in Haarnoja et al. (2018). Concretely, given a constraint value Λ, at each epoch we minimize:

$$J(\lambda) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \mathcal{D}} \left[\log \lambda (\Lambda - \lambda \cdot u(\mathbf{s}_t, \mathbf{a}_t)) \right] \tag{5}$$

We start from an initial weight $\lambda = 1$. We observe that the penalty weight found by automatic tuning tends to converge within the first 50 epochs and then remains stable throughout training.

Table 8: Improvement over grid-searched MOPO through restricted hyperparameter choices (e.g., one single choice, or an $\arg\max$ between two) on the D4RL MuJoCo benchmark. The single and two setup approaches both use the Ensemble Std. penalty and $N = 10$.

Algorithm	Average Score	$\mathbb{P}$[Improvement over MOPO]
MOPO (default hyperparameters)	34.2	-
Single setup: $(h = 20, \Lambda = 1)$	49.0 (+43%)	73.96%
Two setups: $\arg\max\{(h = 10, \Lambda = 0.5), (h = 20, \Lambda = 1)\}$	57.8 (+69%)	80.20%
Optimized MOPO (ours, Table 3)	65.2 (+91%)	89.06%

J BEST FOUND ADROIT HYPERPARAMETERS

We present the best found hyperparameters under our BO procedure for the D4RL Adroit tasks in Table 4. We see similar trends as in our main evaluation in Table 3 favoring higher rollout lengths and the Ensemble penalties.

Table 9: Best discovered hyperparameters using BO for Adroit

Environment		**Discovered Hyperparameters**			
		N	**λ**	**h**	**Penalty**
pen	cloned	10	6.64	12	Ensemble Std
	human	11	0.96	37	Ensemble Var
	expert	7	4.56	5	Max Aleatoric
hammer	cloned	10	0.21	12	Ensemble Var
	human	13	2.48	47	Ensemble Std
	expert	12	0.99	37	Ensemble Std

E

Appendices of Chapter 7

Supplementary Material

Table of Contents

A Offline Dataset Characteristics

We provide explicit statistics on the returns of each episode for the datasets used in our main evaluation. This provides a reasonable proxy to how diverse each dataset is.

Table 5: Full summary statistics of per-episode return in the V-D4RL benchmark.

Dataset		Timesteps	Mean	Std. Dev.	Min.	P25	Median	P75	Max.
walker	random	100K	42.3	8.7	30.0	34.9	41.3	46.8	74.6
	mixed	100K	144.5	155.9	10.9	44.3	69.4	162.4	604.9
	medium	100K	439.6	48.4	176.2	423.1	445.5	466.7	538.0
	medexp	200K	704.1	267.7	176.2	445.5	538.0	969.1	990.6
	expert	100K	969.8	12.4	909.2	963.9	969.1	979.5	990.6
cheetah	random	100K	6.6	2.6	1.1	4.7	6.3	8.4	16.3
	mixed	200K	191.2	144.6	2.5	48.3	191.9	303.4	473.8
	medium	100K	523.8	25.5	325.3	509.1	524.2	538.3	578.3
	medexp	200K	707.0	184.9	325.3	524.2	578.3	894.1	905.7
	expert	100K	891.1	11.2	843.0	886.9	894.2	898.5	905.7
humanoid	random	100K	1.1	0.8	0.0	0.5	1.0	1.5	5.7
	mixed	600K	275.7	176.1	0.0	93.7	341.7	423.1	529.0
	medium	100K	573.0	16.7	526.5	560.6	572.9	584.9	609.4
	medexp	200K	715.9	146.1	526.5	572.9	620.6	877.4	889.8
	expert	100K	858.1	42.4	631.8	846.4	877.6	885.5	889.8

We compare this to the LOMPO datasets and find that they are more widely distributed, due to the differing data-collection method.

Table 6: Summary statistics of per-episode return in the LOMPO DMC walker-walk datasets.

Dataset		Timesteps	Mean	Std. Dev.	Min.	P25	Median	P75	Max.
walker	mixed	100K	208.9	144.1	33.3	75.1	172.4	340.5	496.7
	medexp	100K	674.4	92.9	501.3	596.8	679.8	752.5	869.1
	expert	100K	920.6	81.6	17.9	905.9	950.3	957.9	987.0

As we see in Table 6 and Figure 5, the LOMPO walker-walk expert dataset has a standard deviation roughly 8x higher than our expert dataset and has an extremely wide [min, max] range. Furthermore, whilst our medexp dataset is bimodal, the LOMPO medexp dataset's returns are a continuous progression. This reflects that the LOMPO data is sampled from the second half of a replay buffer after medium-level performance is attained, akin to the medium-replay (mixed) datasets.

A.0.1 Broader Issues with Visual Data

Large datasets consisting of images often contain systematic biases, which can damage generalization. The datasets constructed in this paper are all synthetic from simulated reinforcement learning environments. However, as we move towards applying offline RL from visual observations to real-world tasks, it is important to take these potential dangers into account and extend existing work in algorithmic fairness from computer vision to our setting.

B Algorithmic Details

We provide additional details for both algorithms here and indicate where our modifications have been made.

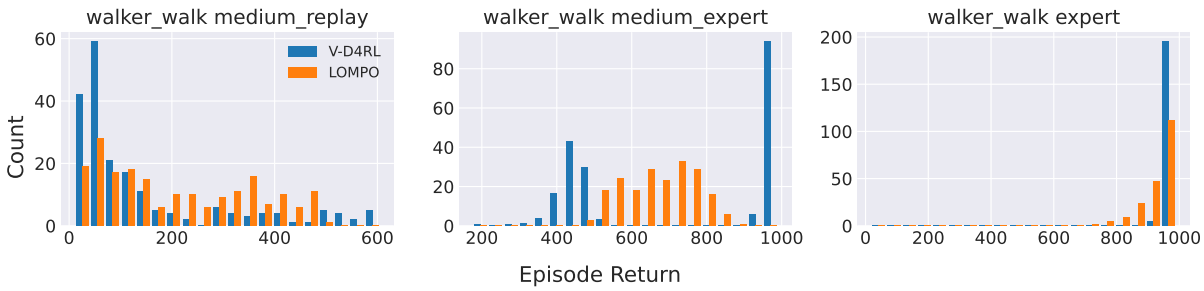


Figure 5: Comparison of the episodic returns from the LOMPO and V-D4RL. We see LOMPO has significantly more diversity in the medium-expert and expert datasets. Note that there is a single episode in the LOMPO expert dataset which has a low return of 17.9.

B.1 Offline DV2

In the offline setting, it suffices to simply perform one phase of model training and one phase of policy training for the DreamerV2 (Hafner et al., 2020b) algorithm. Each episode in the offline dataset is ordered sequentially to facilitate sequence learning. To this instantiation of DreamerV2, we simply add a reward penalty corresponding to the mean disagreement of the dynamics ensemble. During standard DreamerV2 policy training, imagined latent trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ are assigned reward $r_\tau = \mathrm{E}\left[q_\theta\left(\cdot \mid s_\tau\right)\right]$ according to the mean output of the reward predictor. The imagined latent states s_t consist of a deterministic component h_t, implemented as the recurrent state of a GRU, and a stochastic component z_t with categorical distribution. The logits of the categorical distribution are computed from an ensemble (with input h_t) over which we compute the mean disagreement.

B.2 DrQ+BC

Here, we simply modify the policy loss term in DrQ-v2 (Yarats et al., 2021a) to match the loss given in Fujimoto & Gu (2021). Following the notation from Yarats et al. (2021a), the DrQ-v2 actor π_ϕ is trained with the following loss:

$$\mathcal{L}_\phi(\mathcal{D}) = -\mathbb{E}_{\boldsymbol{s}_t \sim \mathcal{D}}\left[Q_\theta\left(\boldsymbol{h}_t, \boldsymbol{a}_t\right)\right]$$

where $\boldsymbol{h}_t = f_\xi\left(\mathrm{aug}\left(\boldsymbol{s}_t\right)\right)$ is the encoded augmented visual observation, $\boldsymbol{a}_t = \pi_\phi\left(\boldsymbol{h}_t\right) + \epsilon$ is the action with clipped noise to smooth the targets $\epsilon \sim \mathrm{clip}\left(\mathcal{N}\left(0, \sigma^2\right), -c, c\right)$. Note that we also do not update encoder weights with the policy gradient. We also train a pair of two fully connected Q-networks, which both use features from a single encoder, and take their minimum when calculating target values and actor losses. The resultant algorithm can be viewed as TD3 (Fujimoto et al., 2018), but with decaying smoothing noise parameters c.

In DrQ+BC, this loss becomes:

$$\mathcal{L}_\phi(\mathcal{D}) = -\mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{a}_t \sim \mathcal{D}}\left[\lambda Q_\theta\left(\boldsymbol{h}_t, \boldsymbol{a}_t\right) - \left(\pi_\phi\left(\boldsymbol{h}_t\right) - \boldsymbol{a}_t\right)^2\right]$$

where $\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(h_i, a_i)} |Q(h_i, a_i)|}$ is an adaptive normalization term computed over minibatches. α is a behavioral cloning weight, always set to 2.5 in Fujimoto & Gu (2021), which we also adopt. We experimented performing an extensive grid search over α, but did not observe any noticeable benefit in our offline datasets by deviating away from the default value.

B.3 CQL

Our CQL implementation was also built on top of the DrQv2 codebase for comparability. We use the CQL($\mathcal{H}$) objective with fixed weight, as we found this to be the most performant. This corresponds to choosing the KL-divergence to a uniform prior as the regularizer $\mathcal{R}(\mu)$ in Kumar et al. (2020). Concretely,

the Q-function objective becomes

$$\min_Q \alpha_{\text{CQL}} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a})) - \mathbb{E}_{\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \right] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}} \left[\left(Q - \hat{\mathcal{B}}^{\pi_k} \hat{Q}^k \right)^2 \right]$$

where α_{CQL} is a trade-off factor, $\hat{\pi}_\beta$ refers to the empirical behavioral policy and $\hat{\mathcal{B}}^{\pi_k}$ to the empirical Bellman operator that backs up a single sample. This is approximated by taking gradient steps and sampling actions from the given bounds.

C Hyperparameter and Experiment Setup

C.1 Offline DV2

Our Offline DV2 implementation was built on top of the official DreamerV2 repository at: `https://github.com/danijar/dreamerv2` with minor modifications. The code was released under the MIT License. Table 7 lists the hyperparameters used for Offline DV2. For other hyperparameter values, we used the default values in the DreamerV2 repository.

Table 7: Offline DV2 hyperparameters.

Parameter	Value(s)
ensemble member count (K)	7
imagination horizon (H)	5
batch size	64
sequence length (L)	50
action repeat	2
observation size	[64, 64]
discount (γ)	0.99
optimizer	Adam
learning rate	{model $= 3 \times 10^{-4}$, actor-critic $= 8 \times 10^{-5}$}
model training epochs	800
agent training epochs	2,400
uncertainty penalty	mean disagreement
uncertainty weight (λ)	in [3, 10]

We found a default value of $\lambda = 10$ works for most settings. The only settings where this changes are $\lambda = 3$ for both random datasets and $\lambda = 8$ for the walker-walk mixed dataset.

For the penalty choice, we chose mean disagreement of the ensemble because it comprises one half of the ensemble variance, which was shown to be an optimal choice for offline model-based reinforcement learning in Lu et al. (2022). We found that the other component of the ensemble variance, the average variance over the ensemble, was uninformative and so discarded it.

C.1.1 Understanding the impact of hyperparameters

We run additional experiments to understand whether the key default online hyperparameters perform well in the offline setting; for proprioceptive environments, this is not always the case (Lu et al., 2022). Using V-D4RL we are able to better understand the impact of these design choices, and we show the results in Table 8. Concretely, the optimal hyperparameters are very different between Offline DV2 and the online DreamerV2. In the offline setting, it is better to increase in batch size from 16 to 64, and a decrease the imagined horizon (H) from 15 to 5. With our setup, the increase in batch size takes around 76% more wall-clock time per batch, but this leads to a huge gain in performance when normalizing for the actual amount of data that is trained on. We believe the reduction in horizon is required due to the lack of online 'remedial' sampling (i.e., sampling data from the true environment that corrects for errors in the imagined rollouts),

thus resulting in exploitation when training solely on the offline data, even with an uncertainty penalty. However, as was also observed in Lu et al. (2022), choosing too low a horizon (i.e., H = 2) is ill-advised, as this results in over-reliance on the offline state distribution, with less chance of policy improvement. Consequently, we choose batch_size = 16 and H = 5 for all our experiments.

Table 8: Ablations on hyperparameters of Offline DV2 where they differ from the online DreamerV2. We report final performance mapped from $[0, 1000]$ to $[0, 100]$ averaged over six seeds.

Environment		Default	batch_size=16	H=2	H=15
walker	random	28.7 ±13.0	22.6 ± 8.4	31.9 ± 7.6	16.9 ± 8.5
	mixed	56.5 ±18.1	27.4 ±16.5	54.7 ±14.5	45.5 ±15.1
	medium	34.1 ±19.7	33.3 ±21.4	47.8 ±18.8	21.0 ±20.3
	medexp	43.9 ±34.4	24.5 ±30.1	22.4 ±18.1	22.9 ±25.7
	expert	4.8 ± 0.6	2.7 ± 1.0	3.4 ± 0.6	3.1 ± 0.8
cheetah	random	31.7 ± 2.7	29.3 ± 4.0	29.8 ± 4.3	32.4 ± 3.7
	mixed	61.6 ± 1.0	53.7 ± 6.9	56.5 ± 2.7	61.3 ± 2.8
	medium	17.2 ± 3.5	17.4 ± 8.0	17.2 ± 6.2	14.3 ± 4.7
	medexp	10.4 ± 3.5	9.4 ± 5.3	8.1 ± 3.8	8.7 ± 5.2
	expert	10.9 ± 3.2	11.1 ± 5.2	8.1 ± 4.8	9.7 ± 3.7
Average Change		-	-21.6%	-6.1%	-15.8%

C.2 DrQ+BC

Our DrQ+BC implementation was built on top of the official DrQ-v2 repository at: `https://github.com/facebookresearch/drqv2`. The code was released under the MIT License. Table 9 lists the hyperparameters used for DrQ+BC. For other hyperparameter values, we used the default values in the DrQ-v2 repository. Due to the size of some of our offline datasets, we found the default replay buffer would not scale to the offline datasets. Thus, we used a NumPy (Harris et al., 2020) array implementation instead.

Table 9: DrQ+BC hyperparameters.

Parameter	Value
batch size	256
action repeat	2
observation size	[84, 84]
discount (γ)	0.99
optimizer	Adam
learning rate	1×10^{-4}
agent training epochs	256
n-step returns.	3
Exploration stddev. clip	0.3
Exploration stddev. schedule.	linear(1.0, 0.1, 500000)
BC Weight (α)	2.5

We also further tuned α within $\{1.5, 2.5, 3.5\}$ but as Fujimoto & Gu (2021) found, we did not observe any noticeable benefit from deviating away from the default value for α.

C.3 Behavioral Cloning

Our BC implementation shares the exact same policy network and hyperparameters in DrQ+BC but just minimizes MSE on the offline data. Consequently, we must also optimize the learned encoder using the supervised learning loss (unlike in DrQ+BC, where the TD-loss only contributes to the encoder representation learning, and not the policy loss).

C.4 CQL

Similarly to BC, our CQL implementation is also based on the same networks and hyperparameters in DrQ+BC. CQL introduces one extra hyperparameter, the trade-off factor α_{CQL}. We perform a hyperparameter sweep for this over the range: $\{0.5, 1, 2, 5, 10, 20\}$. We chose the following values per environment:

Table 10: CQL trade-off factor per environment for Walker and Cheetah. Humanoid omitted as all choices performed equally.

Dataset		Trade-off Factor (α_{CQL})
walker	random	0.5
	mixed	0.5
	medium	2
	medexp	2
	expert	5
cheetah	random	0.5
	mixed	0.5
	medium	10
	medexp	1
	expert	20

C.5 LOMPO

For our LOMPO evaluation, we used the official repository at: `https://github.com/rmrafailov/LOMPO`. The code was open-sourced without license. We perform a hyperparameter search over the uncertainty weight λ in the range $\{1, 5\}$. The default value used in Rafailov et al. (2021) is $\lambda = 5$, but we found that $\lambda = 1$ worked better for random datasets. The accompanying data for the DMC Walker-Walk data from Rafailov et al. (2021) was very kindly provided by the authors.

C.6 Computational Cost

The experiments in this paper were run on NVIDIA V100 GPUs. On the standard V-D4RL 100K datasets, DrQ+BC took 1.6 hours, Offline DV2 took 10 hours, and CQL took 12 hours.

Since we wish to compare several offline algorithms using the same dataset, we define a notion of "offline epoch" for all algorithms to show training performance over time. We simply normalize the total number of gradient steps, so training progress falls within $[0, 1000]$.

D Further Results and Ablation Studies

D.1 Full Tabular Distracted Results

We further present full tabular results from Figure 3 in Section 5.1 for both Offline DV2 and DrQ+BC in Table 11 and Table 12 respectively. The highlighted base unshifted results in both tables are the same as in Table 1 for Offline DV2 and Table 4 for DrQ+BC.

Table 11: Offline DV2 shows surprisingly good generalization to unseen distractions and can handle mixed datasets. Final mean performance is averaged over six random seeds and base undistracted performance is highlighted. The environment used is walker-walk random.

Shift Severity	% Shifted	Evaluation Return		
		Original	Distraction Train	Distraction Test
low	0%	28.7	25.9	20.0
	25%	28.1	22.0	14.0
	50%	22.5	16.0	17.7
	75%	16.3	21.9	20.1
	100%	29.4	29.7	18.6
moderate	0%	28.7	21.0	15.1
	25%	28.6	19.1	14.9
	50%	19.8	20.5	11.2
	75%	24.4	20.0	11.9
	100%	20.1	22.9	15.5
high	0%	28.7	14.1	10.7
	25%	24.6	10.1	6.4
	50%	10.7	19.1	6.0
	75%	20.3	18.9	5.54
	100%	3.2	25.4	5.2

Table 12: DrQ+BC can adapt to multiple different distractions but is extremely brittle to settings it has not seen and struggles to generalize. Final mean performance is averaged over six random seeds and base undistracted performance is highlighted. The environment used is cheetah-run medexp.

Shift Severity	% Shifted	Evaluation Return		
		Original	Distraction Train	Distraction Test
low	0%	79.1	0.5	1.1
	25%	73.9	77.5	8.7
	50%	72.3	82.2	8.5
	75%	67.6	85.5	7.6
	100%	4.2	86.6	4.5
moderate	0%	79.1	0.2	0.9
	25%	74.6	77.9	3.1
	50%	68.0	83.6	3.4
	75%	55.4	86.1	3.7
	100%	0.9	86.7	2.0
high	0%	79.1	0.1	0.5
	25%	76.7	76.8	1.3
	50%	66.7	82.5	1.1
	75%	48.4	84.8	1.2
	100%	0.4	86.1	0.9

D.2 Further Offline DV2 Results on Multitask Datasets

To complement the results in Table 3, we show additional results for Offline DV2 on medexp multitask data. Here, Offline DV2 learns a slightly reduced quality policy compared to that on the base environment, but experiences no deterioration in performance on the test environments in walker or cheetah.

Table 13: Evaluation on the DMControl-Multitask benchmark using *medexp* data for Offline DV2. Normalized performance from [0, 1000] to [0, 100] is averaged over six seeds. Offline DV2 shows a strong ability to generalize to the extrapolation test environments.

Algorithm	**Environment**	**Eval. Return**		
		Train Tasks	**Test Interp.**	**Test Extrap.**
Offline DV2	walker	23.2	16.5	19.8
	cheetah	8.2	7.2	9.6

D.3 DrQ+BC Random-Expert Ablation Studies

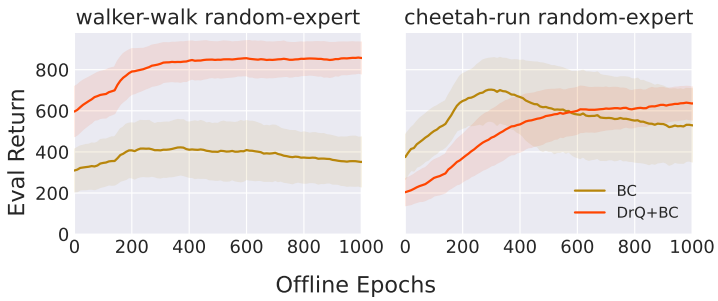


Figure 6: Comparison of DrQ+BC and BC on random-expert datasets with 500,000 of each datatype, results averaged over six seeds.

Analogously to Fujimoto & Gu (2021), we continue to see that DrQ+BC does well on mixed datasets with high reward with experiments on the concatenated random-expert datasets. Surprisingly, we see behavioral cloning also does reasonably well on cheetah-run with random-expert data. This is likely to be because there is significant distribution shift between the states of random and expert trajectories for that environment. This would lead to minimal destructive interference between similar states which contain completely different actions, as these state distributions largely do not overlap. This is in contrast to the medium-expert dataset, which experiences higher state overlap between its two modes (i.e., states generated by a medium and an expert policy respectively), resulting in a marginal “average" action being learned which is likely suboptimal, as can be seen by the poor performance of the BC agent in Tables 1 and 4.

D.4 Analysis on World Models for Visual Observations

In this section, we seek to better understand the differences between model-based and model-free algorithms by presenting further analysis for the Offline DreamerV2 algorithm. In particular, we begin to address our open questions and understand why Offline DreamerV2 deals well with unseen distractions but has trouble handling larger or more narrowly concentrated datasets.

D.4.1 Model Training Time

One of the major factors preventing algorithms that use an RSSM like Offline DV2 and LOMPO from scaling to larger datasets is the time required for model training. The standard number of epochs of model training for our 100,000 datasets in Section 4 is 800 epochs, which takes around six hours on a V100 GPU. This scales

linearly with the number of training points if we maintain the same batch size. As we show in Figure 7, this is mandatory for performance and is a fundamental limitation of current model-based methods. We can see that the evaluated return increases and becomes more tightly distributed as the number of training epochs increases up to 800.

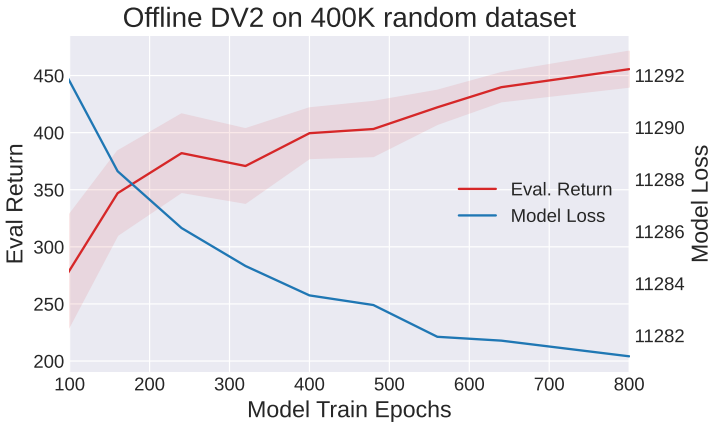


Figure 7: Evaluated return against model train epochs for Offline DV2 on a random dataset of size 400,000. We train a single model up to 800 epochs and evaluate the model periodically on three random seeds. We see that full model training, which scales linearly with training points, is mandatory for good performance.

D.4.2 Uncertainty Quantification

In Table 1, we see that Offline DV2 is considerably weaker on the expert datasets that have narrow data distributions. On these expert datasets, we found that the model uncertainty often had lower variance across a trajectory compared to models trained on other datasets, and was thus uninformative. We give summary statistics for the uncertainty penalty on states generated by a random policy in the walker-walk environment in Table 14. Since we followed the author's DreamerV2 implementation (Hafner et al., 2020b), only the stochastic portion of the latent state is predicted by the ensemble; this may mean crucial calibration is lost when ignoring the impact of the deterministic latent. Future work could involve investigating SVSG (Jain et al., 2021), an extension to DreamerV2 with a purely stochastic latent state.

Table 14: Mean and standard deviation of the uncertainty penalty computed over 1,024 states sampled from the 'random' dataset on the walker-walk environment. Note that the model trained on expert data reports considerably smaller and tighter uncertainty values (compared to 'medium' and 'medexp'), despite the large distribution shift that exists from 'random' to 'expert' data. Instead, we'd expect the 'expert' trained model to exhibit the largest mean uncertainty when tested on the 'random' data.

Dataset Type	**Mean**	**Std.**
random	0.223	0.040
mixed	0.226	0.035
medium	0.341	0.034
medexp	0.338	0.034
expert	0.262	0.020

D.5 Understanding Model-Based Extrapolation

Finally, we investigate the reasons why our model-based baseline, Offline DV2, appears to generalize to unseen distractions as seen in Section 5.1. The RSSM includes a latent decoder for the visual observations, which is primarily used during training to provide a self-supervised reconstruction loss. During deployment, the

policy solely relies on the latent states from the encoder, and the decoder is effectively discarded. However, we can still use the decoder at test time in order to understand the information contained in the latent states. To do this, we take the latent states generated during rollouts in the extrapolation experiments, and 'translate' them into natural images by passing them through the decoder.

We first investigate the setting where each RSSM is trained only on data from a single distraction and then transferred. This setting is the most successful, as can be seen in Figure 3. Thus, in Figure 8 we first show the ground truth observation provided to the agent, then below this we show the decoder output reconstructed from the latent. In many cases, we can recover the original pose despite ending up in different visual surroundings. This is an indication that despite the decoder overfitting to the exogenous factors in the visual input (e.g., background, colors), the latent captures the salient *state* information of the agent (e.g., joint positions and angles), explaining the strong test-time transfer performance.

In Figure 3, we note that settings with a mixture of distractors often had worse test-time transfer performance than those with a single distractor. To explain this, we consider an RSSM trained on images with a combination of two fixed distractors, and examine the reconstruction of an episode under a third distraction. We see in Figure 9, that the RSSM latents are split between the two modes of the data and the reconstruction switches robot color and background midway. This significantly confuses the recovered pose of the walker and likely causes a degradation in performance. Disentangling the factors of variation (Burgess et al., 2018; Mathieu et al., 2019) represents important future work for offline RL from visual observations.

Figure 8: Reconstruction of random episodes shifted by a fixed distractor by models trained on data shifted by a different distractor. The top row shows the original ground truth, and the bottom two rows the model reconstruction for a different RSSM. We can see that in many cases, the original pose of the walker is still able to be recovered.

Figure 9: Bottom row shows the reconstruction of a distracted random episode (on the top row) using an RSSM trained on a mixture of two differently distracted environments. We can see that the reconstructed states are split between the two modes of the data and the reconstruction switches robot color and background midway.