Offline Reinforcement Learning and World Models

Uljad Berdica

Wadham College

University of Oxford

DPhil Transfer Report

November 14, 2024

PART I: Papers

Reinforcement Learning Controllers for Soft Robots Using Learned Environments

IEEE Robosoft 2024

Reinforcement Learning Controllers for Soft Robots using Learned Environments

Uljad Berdica[1,2], Matthew Jackson[1,2], Niccolò Enrico Veronese[3], Jakob Foerster[1], Perla Maiolino[1,2]

***Abstract*— Soft robotic manipulators offer operational advantage due to their compliant and deformable structures. However, their inherently nonlinear dynamics presents substantial challenges. Traditional analytical methods often depend on simplifying assumptions, while learning-based techniques can be computationally demanding and limit the control policies to existing data. This paper introduces a novel approach to soft robotic control, leveraging state-of-the-art policy gradient methods within parallelizable synthetic environments learned from data. We also propose a safety oriented actuation space exploration protocol via cascaded updates and weighted randomness. Specifically, our recurrent forward dynamics model is learned by generating a training dataset from a physically safe *mean reverting* random walk in actuation space to explore the partially-observed state-space. We demonstrate a reinforcement learning approach towards closed-loop control through state-of-the-art actor-critic methods, which efficiently learn high-performance behaviour over long horizons. This approach removes the need for any knowledge regarding the robot's operation or capabilities and sets the stage for a comprehensive benchmarking tool in soft robotics control.**

***Index Terms*— soft manipulator, reinforcement learning, learned controllers, simulators**

I. Introduction

Soft robotic manipulators are made of compliant material and exhibit a low Young's modulus that enables them to be arranged in highly deformable geometries [1]. These designs, inspired by biological organisms, can undergo large elastic deformation throughout operations and facilitate safer interaction with the environments compared to their traditional rigid counterparts [2]. The morphological dexterity outsources parts of the solution computation to the compliant material [3], but remains underactuated as the states of the physical body are governed by highly nonlinear continuum dynamics. Given the inherent challenges, the precise control of soft robots remains an open problem.

The existing analytical methods for accurate dynamic models in classical optimal control make reductive assumptions like constant-curvature and valve control heuristics for trajectory optimization [2] while relying on the material properties being unchanged or otherwise predictably modeled. These methods strive to reduce the computational complexity of the dynamic model while not suppressing the modeling of the adaptive behavior that emerges through the soft robot's interaction with the environment. Moreover, parametric models fail to capture the computation embodied in the morphology of the robot, making data-driven models necessary for capturing the important insight from resulting deformations [4].

Deep learning-based approaches utilize platforms with internal sensory data like pressure and Inertial Measurement Units (IMU) [5] and external sensory data like visual trackers [6] from the robot's interaction with the environment. This allows for the learned models to make use of the morphological changes that occur in operation time. However, the use of learned (black-box) mappings between actuation and task space comes at an increased computational cost both during training and at test time. The most commonly used architecture for such mapping is a non-linear autoregressive network with exogenous inputs (NARX) with one [6] to four [7], [8], [9] time delays. We also employ a recurrent architecture in the form of long short-term memory (LSTM) [10] as implemented in [11] for faster training and inference time. Both NARX and LSTM architectures are designed to learn from distant interactions by overcoming the gradient vanishing problem. While NARX networks tackle this problem through delayed connection from distant past, the contribution is small and scales the computation by a factor equal to that of time-delayed connections [12]. Considering entire sequences of actuations and observations is essential for learning closed loop control of soft robots that is not privy to and limited by prior knowledge of their dynamics.

Reinforcement Learning (RL) algorithms have been successful in solving sequential decision making problems under these limitations by learning through repeated interactions with the environment which in this work is represented as a recurrent model trained from collected data on the robot. Popular success stories include Proximal Policy Optimization (PPO) [13], which has been particularly successful in continuous control [14], [15]. The adoption of deep learning based methods in RL, their large computational requirements, and algorithmic complexity has caused an explosion of different frameworks. These frameworks aim to balance high performance hardware utilization and ease-of-use for rapid prototyping [16]. A recent breakthrough is PureJaxRL [17], which uses JAX [18] to run agents and environments jointly on the GPU, resulting in order of magnitude speedup compared to prior approaches.

Previous work on the use of RL for the control of soft includes value-based methods in [19], [20] and with early attempts of actor-critic methods in [21]. The most recent

[2] Supported by Autonomous Intelligent Machines and Systems (EPSRC Centre for Doctoral Training) EP/S024050/1 and EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1)

[1]University of Oxford {uljad.berdica, matthew.jackson, perla.maiolino, jakob.foerster}@eng.ox.ac.uk

[3] Polytechnic University of Milan, Italy

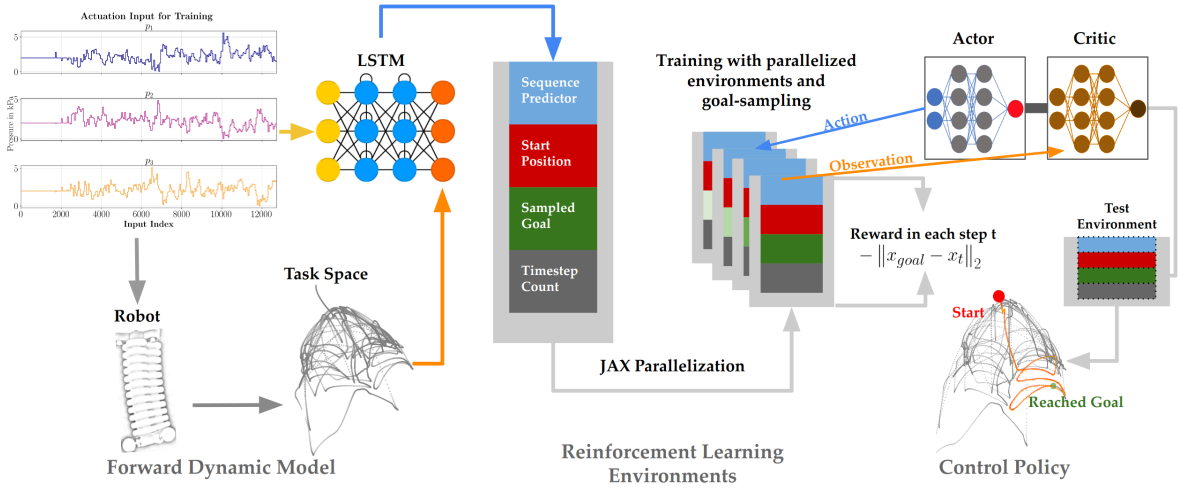


Fig. 1: The pipeline of the learned environment-based solution proposed in this work. The recurrent network to the left represents the LSTM at the core of the synthetic environments.

work to date using closed loop policy gradients leverages Cosserat Rod Models simulations to learn the forward dynamics [8]. These methods hinge on the limited number of recorded interactions in the dataset, prior knowledge of material heuristics [19] or of guiding trajectories for the policy search [22], [6]. We propose a methodology based on forward dynamic models learned in absence of simulations or guiding trajectories.

This work seeks to bring together the advantages of learning-based solutions to soft robotics control by utilizing SOTA PPO implementations [17] to learn closed loop controllers inside environment models implemented via high-performance computing libraries [23]. Our method bypasses the need for any analytical models or prior information regarding the robot's operation. The forward dynamics model is learned by training with the data collected on the robot through a mean reverting random walk in actuation space. Feedforward and recurrent control policies are learned by interacting with the parameterized forward dynamics model wrapped in a JAX-based environment.

In this paper, we first introduce the methods used to collect that data from the robot with examples of the generated input sequences. We then describe the architecture and training process for the supervised learning of the forward dynamics model through a sequence-to-sequence (seq2seq) prediction model as well as the policy optimization procedure to train the the actor-critic network. Finally, we present the training results and inference examples of the closed-loop policies conditioned on observations and tested in the forward dynamic model.

II. Methods

Figure 1 shows the full pipeline implemented in this work starting from the state space exploration to the left (II-A), the creation of the dataset (III-B) and the training of the forward dynamic model (II-B) using a LSTM network. We leverage JAX to generate parallel environments II-C from the trained LSTM model and actor-critic network to simultaneously learn from multiple training trajectories sampled directly from a task space distribution to update policies without depending on previous example trajectories(II-D). This method enables batching multiple goals for robust policy learning. Finally consecutive goal learning is shown by selecting a desired target position for which an action sequence is generated.

A. State Space Exploration

The sequence-to-sequence (seq2seq) mapping from actuation space to task space requires sufficient exploration to reproduce a representative environment interaction for the online actor-critic training. The exploration in this work does not require any static workspace assumptions and is exclusively in the actuation space.

The single constraint to this exploration policy is the maximum pressure P_{max} allowed across the valves of the robot to ensure the full functionality and structural integrity of the soft robot. The protocol to cover a wide range of robot configurations by applying p_j actuation pressures within the safety value uses a mean reverting random walk [24] with tunable parameters, specifically a sigmoid scaling $1/(1+e^{-x})$ with two parameters α and β to account for the randomness and the position of the sigmoid inflection point respectively. α weights the previous input's effect on the next one whereas β modulates the average value of the total pressure. Each generated term p^*_{i+1} of each valve is incrementally added to the previous pressure p^*_i with the generated terms being normally distributed around the preloaded pressure value p_b of the robot in the resting state. Eq. 1 describes the cascading update procedure to keep the sum of p_j below P_{max} in every iteration i.

$$\begin{aligned} &\forall i \in \{0,1,\ldots,N\},\ \forall j \in \{0,1,\ldots,N_{valves}\} \\ &p^*_{i+1,j} = \alpha\ p_{i,j} + (1-\alpha)\ \mathcal{N}(p_b, 1) \\ &p_{i+1} = P_{max}\ \sigma\left(\beta \sum_j p^*_{i,j}\right) \frac{p^*_{i+1,j}}{\sum_j p^*_{i,j}} \end{aligned} \tag{1}$$

- i indexes the iteration,
- j indexes the valves,
- N is the total number of iterations,
- N_{valves} is the total number of valves.

B. Forward Dynamic Model Learning

We use an LSTM to learn the dynamic mapping between the actuation and task space. This has been chosen to be able to train for significantly longer sequences. To increase the predictive versatility of the learned environment around the LSTM latent states, the testing pairs are generated by using a sliding window approach with a step of one in permuted order. Each training pair in the dataset contains the three actuation pressures and the corresponding robot reference point in Cartesian coordinates. Each sequence consists of 512 steps. This number of steps was chosen as it is on average 100 steps higher than the mean reverting random walk procedure which allows to record the initial preload pressure state and the return to that initial state upon the end of the exploration. The length of these exploratory runs is dependent on the limitations of physical platforms used in section III-A.

This is illustrated in Fig. 2 for the x-direction and the actuation pressure p_1 where the training pairs are shown with matched colors. The sliding sequence method effectively places every data points in every possible context of the sequence during training time which allows for context-independent prediction of outputs and the learning of behaviors that were not seen in random exploration like reaching unseen targets as the results in sec IV show.

The model was trained by dividing the dataset in training and testing with a 75-25% split. To improve the generalisation capability of the model, we consider two approaches for the order of passing the data to the model in training. One is to feed the generated sequences as they appear in the dataset to maintain as much of the history of the system as possible. The other approach is to randomly permute the order in which the sequences are passed through the network meaning that temporally close sequences can be seen by the training network at time steps that are further apart than the time difference of their occurrence in the data. The empirical observations in section IV-A show that the random permutation of the sequence pairs performs better with test sequences that were not seen during training.

C. Generation of Reinforcement Learning Environment

The RL agent only interacts with multiple instances of an environment learned from an offline dataset. The environments in this paper are chosen as platforms to interface with the learned dynamic model in a more structured way that allows the policy network to be trained for different goals in each episode that are sampled from the task space i.e. Cartesian coordinates range of the collected data.

As mentioned in sec I, we leverage JAX [18] to achieve high-performance training and inference from our learned environment. JAX objects are compiled with XLA and executed in parallel on GPU. This enables both our model and policy to be trained entirely on GPU, leading to huge speedups against CPU-based methods. We implement our model in the Gymnax environment framework [23], allowing existing agent implementations for online RL to be used seamlessly with our model. The Gymnax frameworks implements environments as classes the instances of which can be run in parallel and have different states allowing for different training conditions and rewards.

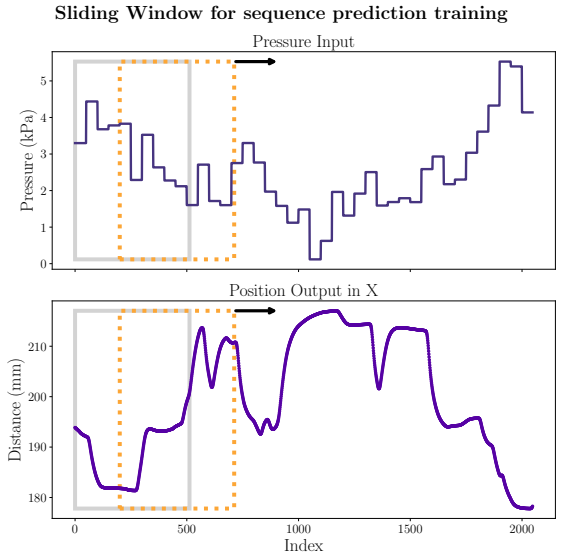


Fig. 2: Training Pair Generation is shown through matched colors. For illustration purposes we use a sequence length of 512 with *step size* of 200 on a subset of the data. In practice, we use a *step size* of 1 and slide through the entire runs.

D. Policy Optimization

We used a policy-gradient methods to train an actor-critic network [16], [13]. Specifically, we use proximal policy optimization (PPO) [13] for policy optimization. PPO uses a trust region approach to stabilise the policy gradient update, employing a clipped surrogate objective which prevents the parameterized policy diverging too far from its original value after it is updated. Policy methods are preferred over value-based method for this robotics application due to the constrained policy update when optimizing for an objective function.

The policy is trained by concatenating the final target to the observations from the environment. If the target destination in task space has been reached within the measurement error distance of 1 mm or if the episode terminates after a predetermined number of steps, the environment resets to the initial state and samples from a range within the reachable dynamic space of the robot before starting the new training episode steps. Note that this is not necessarily a target included on the dataset.

The implementation of the goal perturbation is done via the Algorithm 1. The algorithm takes in the Forward Dynamic Model, the parameters of the environment like starting position, total time steps per episode as well as the training

configuration with the values of the relevant hyperparameters like number of updates, batch size and learning rate. A new goal is set every time the environment state is reset. The output is the parameters of the conditional probability function of an action for a given observation, also referred to as a policy $\pi(\cdot|observation)$ in reinforcement learning terms. Policy π is obtained by training the actor-critic network that learns to predict the actions and values of observations or embedding of observations depending on whether the network is feed-forward or recurrent respectively. Both these networks are trained using PPO [13] implementations based on PureJaxRL [17].

Algorithm 1 Policy Training with Goal Perturbation

Input:
Forward Dynamic Model,
Environment Parameters,
Training Configuration
Output: Parameters of control policy $\pi(\cdot|observation)$
▷ Initialisation of network and environment parameters
$env\_params \leftarrow$ Environment Parameters
$\pi \leftarrow$ initialize actor-critic network parameters
$train\_config \leftarrow$ Training Configuration
▷ create parallel $environment$ with Gymnax [23]
$environment \leftarrow$ Gymnax(env_params)
▷ Loop condition values from Training Configuration
for $t <$ `TOTAL_UPDATES` **or** not converged **do**
  if $t = 0$ **then**
    $observation \leftarrow$ initial observation
    $perturbation \leftarrow env\_params$
    $initial\_pose \leftarrow env\_params$
    $goal \leftarrow \mathcal{N}(0,1) \cdot perturbation + initial\_pose$
  end if
  $actions \leftarrow \pi(observation)$
  $observation \leftarrow$ Forward Dynamic Model($actions$)
  $reward \leftarrow -\left\|goal - observation\right\|_2$
  $\pi \leftarrow$ PPO($reward, goal, train\_config$) [17]
  $t \leftarrow t + 1$
end for
Output: $\pi(\cdot|observation)$

III. Experimental Setup

A. Robot

To validate our approach we use a soft a three chambers bellow-shaped actuator connected to a rigid frame in Fig. 3. The actuation is achieved with compressed air controlled by three separate proportional Festo valves (VEAA-L-3-D2-Q4-V1-1R1). The number of controllable inputs corresponds to the number of chambers, therefore, N_{valves} is 3 (see eq. 1 in section II-A). Reflective markers are integrated at the top and bottom of the actuator to track its movements. The position of the robot reference point in Cartesian coordinates is estimated as the centroid of the triangle marked by the bottom grey reflective markers in Fig. 3 using an Optitrack Motive system equipped with four Flex 3 Cameras. The valves are controlled within a ROS2 [25] environment.

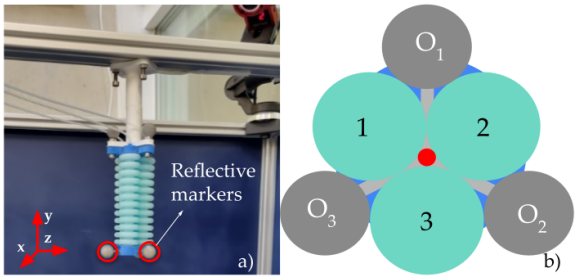


Fig. 3: Robot at initial positions (a) no deformation at home position with initial baseline pressure 2kPa, (b) Transverse cross-section view of the root, pressure chambers A to C and reflective markers O_1 to O_3

B. Dataset Preparation

The mean reverting random walk under the pressure constraints from section II-A is implemented with a P_{max} of 13 kPa, a p_b of 2kPa, N_{valves} of 3 and N of 50. Fig. 4 shows the generated input pressure action sequences for different values of α. An N of 50 iterations was chosen for clarity of visualization to demonstrate that the inputs are independent and diverse while not exceeding the safety limit of P_{max}.

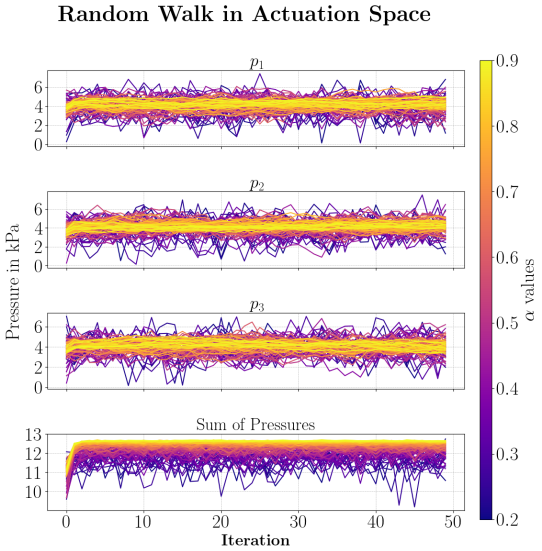


Fig. 4: Random Walk in actuation space for different exploration hyperparameter α

Fig. 5 shows the resulting task space trajectories as a result of the exploration method implemented for different α values of randomness.

The control pressure data is collected through the ROS2 interface and the consecutive positions of the markers are collected with the Optitrack. The Optitrack measurements are acquired at a frequency higher than that of the pressure measurements. To make sure every measured Cartesian coordinate corresponds to a pressure measurement for the

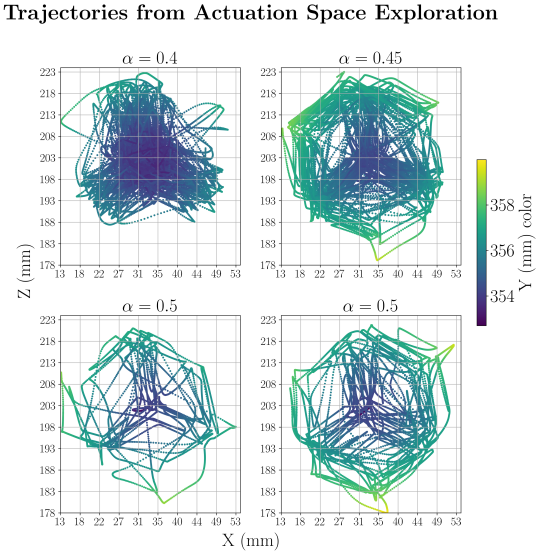


Fig. 5: Resulting trajectories from a random walk in actuation space various levels of randomness

training with the sequences of pairs described in II-B, we match every pressure value to the nearest position in time through a nearest neighbors search through the timestamps of each Optitrack measurement. The rows in the Optitrack measurements that do not get matched to a pressure value are filled with the earliest possible pressure value as the pressure in the robot remains the same before it is changed in a step-like manner. The control frequency is kept at a constant of 2Hz to accommodate to the physical limitations of the setup and enable the covering of the whole dynamic motion range of the robot.

IV. Results

A. Forward Dynamic Model

Fig. 6 shows the training and test results for the mapping from actuation to cartesian positioning via the collected data. The two approaches to passing the training dataset through the model described in section II-B were implemented and tested on the same set of sequences held out during training. In $2 \cdot 10^5$ training steps, it becomes clear that the general approach of permuting the order of temporally close sequences achieves higher train and test results than the sequential training with the same sequences.

B. Task Space Reconstruction

From Fig. 6 we can see that both training and validation losses for the randomly permuted case are low showing that the model accurately predict the task space. However, it is necessary to evaluate the ability of the forward dynamic model to reconstruct a correct task space from exploratory input sequences in test time as it is a core requirement for the data efficient development of closed loop control policies via learned environments.

We evaluate this aspect by qualitatively comparing the task space reconstruction from data seen during the training and

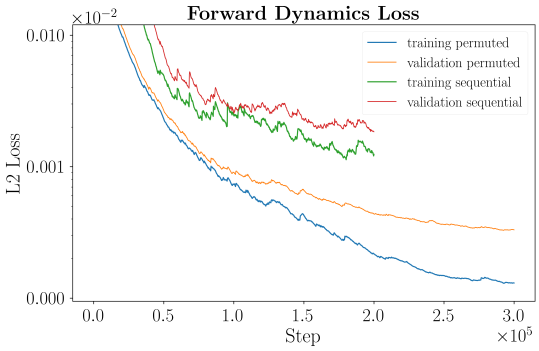
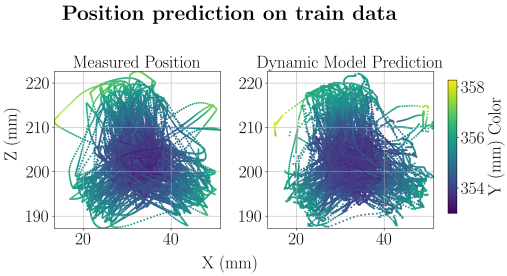
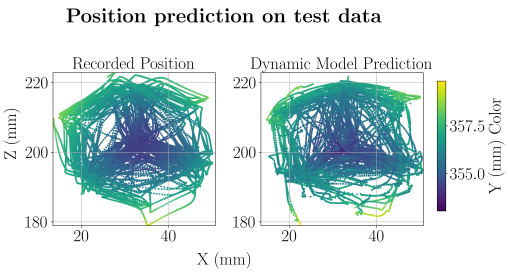


Fig. 6: Training and Validation Losses for the Forward Model. The plots are smoothed using an exponential moving average with a factor of 0.025

data from previously unseen sequences. Fig. 7a shows a close reconstruction of training data and Fig. 7b shows a close reconstruction of an entire run that has not been used in training. These results set the ground for the use of learned models in RL environments.



(a) Reproduction of training data as seen on top left of Fig. 4



(b) Reproduction of unseen action space exploration sequence

Fig. 7: Predictive performance of the model on training and testing datasets.

C. PPO on the Learned Environment

To evaluate the impact of the robots movements history in the episode we implemented two different policies. Specifically one conditioned to the latest observation and one conditioned to the entire history of observations (recurrent).

Fig. 8 shows the rewards monotonically increase with the number of episodes. This also further validates this result as the rewards is monotonically increasing to convergence within 3mm of the target. The mean with one standard de-

viation shaded region obtained by running with 20 different random seeds is plotted for each policy type.

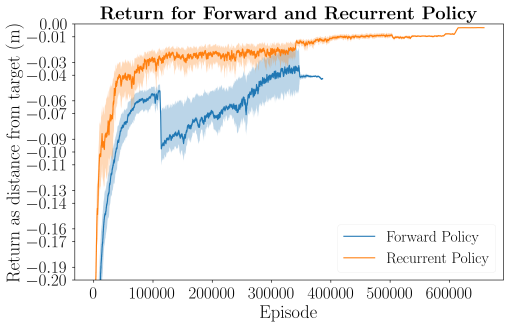


Fig. 8: Episodic return for Forward and Recurrent Policies with one standard deviation shaded region for 20 different random seeds

V. Conclusion

In this paper, we use a model-free approach to learn the forward dynamics of a soft robotic arm. We develop a protocol for state space exploration using random walks and use the generated data to train our model. We demonstrate the effectiveness of our approach in recreating tasks from test sequences and show its potential for developing closed-loop control policies in soft robotics. This general methodology for developing a closed loop control policies shows great promise towards establishing new soft robotics control benchmarks and bridging the physical advantages of soft robotics with the most recent work in Machine Learning. The demonstrated ability of synthetic environments to facilitate planning on real-world data can provide a path towards future work in data-driven emergence of complex behavior, learned sim2real adaptation strategies and further testing of such policies on physical robots with generated actuation regimes beyond general exploration.

References

[1] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science robotics*, vol. 1, no. 1, p. eaah3690, 2016. 1

[2] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018. 1

[3] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological cybernetics*, vol. 105, pp. 355–370, 2011. 1

[4] C. Laschi, T. G. Thuruthel, F. Lida, R. Merzouki, and E. Falotico, "Learning-Based Control Strategies for Soft Robots: Theory, Achievements, and Future Challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, pp. 100–113, June 2023, conference Name: IEEE Control Systems Magazine. 1

[5] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2018, pp. 39–45. 1

[6] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, Feb. 2019, conference Name: IEEE Transactions on Robotics. 1, 2

[7] ——, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & biomimetics*, vol. 12, no. 6, p. 066003, 2017. 1

[8] C. Alessi, H. Hauser, A. Lucantonio, and E. Falotico, "Learning a controller for soft robotic arms and testing its generalization to new observations, dynamics, and tasks," in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2023, pp. 1–7. 1, 2

[9] F. Piqué, H. T. Kalidindi, L. Fruzzetti, C. Laschi, A. Menciassi, and E. Falotico, "Controlling soft robotic arms using continual learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5469–5476, 2022. 1

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 1

[11] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2023. [Online]. Available: http://github.com/google/flax 1

[12] R. DiPietro, C. Rupprecht, N. Navab, and G. D. Hager, "Analyzing and exploiting narx recurrent neural networks for long-term dependencies," *arXiv preprint arXiv:1702.07805*, 2017. 1

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. 1, 3, 4

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015. 1

[15] H. Van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement Learning: State-of-the-Art*. Springer, 2012, pp. 207–251. 1

[16] M. Hessel, M. Kroiss, A. Clark, I. Kemaev, J. Quan, T. Keck, F. Viola, and H. van Hasselt, "Podracer architectures for scalable reinforcement learning," *arXiv preprint arXiv:2104.06272*, 2021. 1, 3

[17] C. Lu, J. Kuba, A. Letcher, L. Metz, C. Schroeder de Witt, and J. Foerster, "Discovered policy optimisation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 455–16 468, 2022. 1, 2, 4

[18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, *et al.*, "Jax: composable transformations of python+ numpy programs," 2018. 1, 3

[19] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, "Open loop position control of soft continuum arm using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5133–5139. 1, 2

[20] X. You, Y. Zhang, X. Chen, X. Liu, Z. Wang, H. Jiang, and X. Chen, "Model-free control for soft manipulators based on reinforcement learning," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2909–2915. 1

[21] Y. Ansari, M. Manti, E. Falotico, Y. Mollard, M. Cianchetti, and C. Laschi, "Towards the development of a soft manipulator as an assistive robot for personal care of elderly people," *International Journal of Advanced Robotic Systems*, vol. 14, no. 2, p. 1729881416687132, 2017. 1

[22] M. Rolf, J. J. Steil, and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 216–229, 2010. 2

[23] R. T. Lange, "gymnax: A jax-based reinforcement learning environment library, 2022b," *URL http://github. com/RobertTLange/gymnax*, 2022. 2, 3, 4

[24] Wolfram Demonstrations Project, "Mean reverting random walks," https://demonstrations.wolfram.com/MeanRevertingRandomWalks/, 2011, accessed: 2024-03-01. 2

[25] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. 4

Papers

Investigating Online RL in World Models

Under Review for ICLR 2025

Robust Offline Learning via Adversarial World Models

NeurIPS 2024 Workshop on Open-World Agents

NeurIPS 2024 Workshop on New Frontiers in Adversarial Machine Learning

INVESTIGATING ONLINE RL IN WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Over the past decade, online reinforcement learning (RL) has made drastic improvements in a number of settings, such as video games and robotics. However, despite these successes, the impact of RL on many *real-world* problems has remained limited. Underlying this fact is that, in many settings, we are unable to learn in an online fashion due to excessive cost and safety requirements or lack of an accurate simulator. In principle, foundation world models trained on large-scale uncurated offline data such as internet videos and other modalities could provide a training paradigm for generalist AI agents which alleviates the need for task specific simulation environments. Unfortunately, training inside world models is usually studied in the context of offline RL, where popular datasets have a biased structure. This necessitates short roll-outs or other severely limiting mechanisms to prevent model exploitation. Here we probe under what circumstances full roll-out training inside world models is possible *without* any penalties. We find that on a non-adversarial offline dataset simply ensembling over a large number of independently trained world models is sufficient to ensure transfer to the real world, even for datasets that are orders of magnitude smaller than is common in offline RL. Interestingly, more sophisticated methods for level selection provide no advantage and standard offline RL underperform in this setting. We open source all our code and data to facilitate further work in this direction.

1 INTRODUCTION

Exploiting large amounts of data has proven to be a crucial component of recent advancements in machine learning. Generative models across multiple modalities—such as large language models (e.g., (OpenAI et al., 2024; Touvron et al., 2023)), text-to-image models (e.g., (Imagen-Team-Google et al., 2024; Betker et al., 2023)), and text-to-video models (e.g., (Brooks et al., 2024))—demonstrate that scale and coverage often outweigh the benefits of curation or the injection of favorable biases.

Reinforcement Learning (RL) (Sutton & Barto, 2018) has shown great promise in solving complex problems whenever *fast and accurate* simulation environments are available, such as in computer games (Silver et al., 2016a) and, to a lesser extent, robotics. Unfortunately, reliance on simulators has severely limited the applicability of RL to real-world problem settings. World models (Ha & Schmidhuber, 2018) in principle offer a solution by learning approximate dynamics models from state transitions data. These models are trained in a supervised manner, reducing reliance on task-specific hand-coded simulators. However, while increasing the dataset size can improve the fidelity of learned world models, they are rarely perfect recreations of the underlying environment. Ha & Schmidhuber (2018) demonstrate how RL agents frequently learn to exploit discontinuities and edge cases in *learned* dynamics to receive large spikes in simulated reward while learning unhelpful behaviours for the true environment.

This is a common issue in *offline RL*, where the goal is to produce high-performing policies based only on a *static offline dataset* without further interactions with the real environment. To address this problem, offline RL methods have introduced a number of mechanisms to regularise the learning process towards the offline data distribution and enforce conservatism Kumar et al. (2020). These include severely *truncating rollouts*, which limits the consecutive number of steps an agent is allowed to take inside a world model, and *uncertainty penalties*, which discourage the agent from stepping into parts of the state space where the world model is not confident Kumar et al. (2020).

These mechanisms are particularly relevant for current offline-RL benchmarks like D4RL (Fu et al., 2020), since the datasets are structured to be *adversarial* in terms of data coverage.

However, these mechanisms are clearly limiting if the goal is to train *generalist RL agents* fully inside a large-scale foundation generative world model, which has recently emerged as a possible path towards agentic AI systems (Bruce et al., 2024). Crucially, this approach would allow RL to take advantage of the rapid progress made on generative modeling on the back of large amounts of uncurated data.

In this paper, we investigate to what extent it is possible to *remove* all of the mechanisms introduced to offline RL and instead train agents on full-length roll-outs without any penalty terms *inside learnt models*, while still achieving successful transfer to the real underlying environment. Specifically, we probe two different axes: First off, we produce a dataset that has more uniform coverage (in terms of task performance), since we believe this is more representative of a generalist world model. To account for the fact that there might be limited amounts of task-specific data available, we probe how our method performance scales across various offline-dataset sizes, ranging from a few thousand transitions to millions. Secondly, we investigate whether Unsupervised Environment Design (UED) (Dennis et al., 2020; Jiang et al., 2021b;a; Parker-Holder et al., 2022) can be used to improve real-world transfer of models trained in simulation.

UED is a class of online RL methods that can address the need for zero-shot adaptations by training agents to be robust across varying train and test distributions. These methods seek to minimize maximum regret over a space of levels (Dennis et al., 2020); however, they require convergence to a Nash equilibrium—a guarantee that is not always achievable. Additionally, UED approaches often necessitate specific domain knowledge for parameter tuning and can lead to learning stagnation once regret bounds are met across all configurations (Beukman et al., 2024). Specifically, we independently train a large number (100) of world models (deep neural networks) on our offline dataset and treat each of these models as a given *level* in UED.

Surprisingly, we find that simply doing *domain randomisation* over this level space (i.e. picking a different model each time step or episode) drastically improves the test-time performance across different dataset sizes, while more advanced UED methods offer no significant advantage. Furthermore, "standard" offline RL methods perform poorly in our setting, since they rely heavily on the offline dataset containing task specific high-quality demonstration data and on conservatism. Lastly, while our random ensembling approach avoids catastrophic exploitation of the world model in the popular D4RL benchmark, our method fails to perform well in this setting. Our analysis suggests that this is due to insufficient coverage which was specifically designed to mirror the challenges of offline RL (rather than a generalist world model).

We open-source all our code, including the new dataset, and hope that this line of work will not only encourage practitioners to question some of the assumptions underlying the offline RL literature but also present a first step towards training RL agents on *full trajectories* inside generative models.

2 Preliminaries

2.1 Contextual Markov Decision Process

We define a infinite-horizon, discounted contextual Markov decision process (CMDP) (Hallak et al., 2015) by introducing a context variable $\theta \in \Theta \subseteq \mathbb{R}^d$:

$$\mathcal{M}(\theta) \coloneqq \langle \mathcal{S}, \mathcal{A}, P_0, P_S(s,a,\theta), P_R(s,a,\theta), \gamma \rangle, \tag{1}$$

where each θ indexes a specific MDP by parametrising a transition distribution $P_{\mathcal{S}}(s,a,\theta) : \mathcal{S} \times \mathcal{A} \times \Theta \to \mathcal{P}(\mathcal{S})$ and reward distribution $P_R(s,a,\theta) : \mathcal{S} \times \mathcal{A} \times \Theta \to \mathcal{P}(\mathbb{R})$. We denote the corresponding joint conditional state-reward transition distribution as $P_{R,S}(s,a,\theta)$. Partial observability can be integrated into the contextual RL approach by making θ index a specific partially observable MDP (POMDP). For simplicity, we only consider fully observable MDPs in this paper, although our results hold in the more general partially observable setting too.

At timestep t, an agent follows a policy $\pi : \mathcal{S} \times \Theta \to \mathcal{P}(\mathcal{A})$, taking actions $a_t \sim \pi(s_t, \theta)$. We denote the set of all context-conditioned policies as $\Pi_\Theta \coloneqq \{\pi : \mathcal{S} \times \Theta \to \mathcal{P}(\mathcal{A})\}$. The agent is assigned an initial state $s_0 \sim P_0$. As the agent interacts with the environment, it observes a

history of data $h_t \coloneqq \{s_0, a_0, r_0, s_1, a_1, r_1, \dots a_{t-1}, r_{t-1}, s_t\} \in \mathcal{H}_t$ where $\mathcal{H}_t$ is the corresponding state-action-reward product space. We denote the context-conditioned distribution over history h_t as: $P_t^\pi(\theta)$ with density $p_t^\pi(h_t|\theta) = p_0(s_0)\prod_{i=0}^{t}\pi(a_i|s_i,\theta)p(r_i, s_{i+1}|s_i, a_i, \theta)$.

In the infinite-horizon, discounted setting, the goal of an agent in MDP $\mathcal{M}(\theta)$ is to find a policy that optimises the objective:

$$J^\pi(\theta) = \mathbb{E}_{\tau_\infty \sim P_\infty^\pi(\theta)}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]. \tag{2}$$

We denote an optimal policy as $\pi^\star(\cdot,\theta) \in \Pi_\Theta^\star(\theta) \coloneqq \arg\max_{\pi\in\Pi_\Theta} J^\pi(\theta)$, where $\Pi_\Theta^\star(\theta)$ is the set of all optimal MDP-conditioned policies that are optimal for $\mathcal{M}(\theta)$.

2.2 Unsupervised Environment Design

Unsupervised environment design (UED) is a class of autocurriculum methods for RL, where an adversary proposes tasks for an agent to train on. Commonly (Dennis et al., 2020), environments are modelled as a CMDP $\mathcal{M}(\theta)$ (see Equation (1)) known as underspecified Markov decision process where each context $\theta \in \Theta$ is known as a level.

The recent approach of Minimax Regret (MMR) UED has emerged as a promising way to train robust agents (Dennis et al., 2020; Jiang et al., 2021b;a; Parker-Holder et al., 2022). Here, the adversary chooses levels that maximise the agent's *regret*, defined as:

$$\text{Regret}_\theta(\pi) \coloneqq J^{\pi^\star}(\theta) - J^\pi(\theta). \tag{3}$$

Dennis et al. (2020) posed the UED setting as a two-player, zero-sum game between the adversary and the policy. This setting Furthermore, they showed that if the adversary aims to maximize regret, and it is in Nash equilibrium with the policy, the policy satisfies the following equation:

$$\pi_{\text{MinMax}} \in \arg\min_{\pi\in\Pi_\mathcal{H}}\{\max_{\theta\in\Theta}\{\text{Regret}_\theta(\pi)\}\}. \tag{4}$$

Therefore, the policy's minimizes its worst-case regret. This confers a degree of robustness to the policy, as its regret in any level $\theta \in \Theta$ must be below this bound. See Appendix A.1 for a more detailed discussion.

2.2.1 Prioritized Level Replay

Prioritized Level Replay (Jiang et al., 2021b) is an empirically successful curriculum method that relies on curating high-scoring levels. In practice, PLR maintains a buffer of previous high-scoring levels, and either samples from this buffer, or samples new levels. The agent is rolled out on these new levels, and they are scored depending on its performance. High-scoring levels are added to the buffer, and the agent trains on the collected experience. It is not necessary to sample new levels, however; in some cases, the level set may be predefined and fixed (Tzannetos et al., 2024).

The original PLR scores each level θ_i using a time-averaged L_1 value loss of each agent's last trajectory on the level (Jiang et al., 2021b). In order to achieve *minimax robustness*, a scoring function should account for regret as described in Section 2.2. Jiang et al. (2021a) propose different scoring functions that more closely approximate the regret. Ultimately, the choice of a scoring function is a design choice depending on the nature of the environment. We further elaborate on the scoring function choices in section 3.

2.3 World Models

As defined by Ha & Schmidhuber (2018), world models are compact representations of the dynamics of an environment which are independent of the agent's interactions with it; from the agent's perspective, a trained world model can be interacted with in the same way as the true environment. In this work, we assume the dynamics of the environment are Markovian and use a one-step predictive world model. World models are generally represented using a neural network that jointly parameterises the transition distribution $P_\mathcal{S}$ and rewards distribution $P_\mathcal{R}$ from Equation (1). Therefore, one-step transition dynamics of the environment can be modelled by world model $\mathcal{F}_\theta$ as $\hat{s}_{t+1}, \hat{r}_{t+1} \leftarrow \mathcal{F}_\theta(\hat{s}_t, a_t)$ by predicting *both* the state transition and the reward of the agent.

An agent trained entirely in a world model can also learn to take advantage of out-of-distribution or discontinuous areas of the world model's state space (Levine et al., 2020; Ha & Schmidhuber, 2018). These discontinuities *do not* exist in the true underlying environment, and thus the agent learns a policy which does not perform well in practice. Learned world models, much like other model-based offline RL methods, are also subject to compounding error where the difference between the world model outputs and the environment outputs diverge further as the episode proceeds (Saleh et al., 2022). As a result, agents trained on fully offline long-horizon rollouts often lack robustness in transfer to the real environment. Our methods uses an ensemble of world models trained on the same data, and shows a path towards mitigating these issues by learning a transferable policy through fully offline rollouts.

3 Training with Ensembles of World Models

We introduce a set of approaches aiming to leverage large datasets not curated for particular tasks while still benefiting from methods with strong theoretical guarantees used in online RL methods. As shown in Figure 1, we start by training a collection of world models consistent with the provided data. We then treat these models as *levels* and select them based on different sampling methods to train a robust, transferable policy.

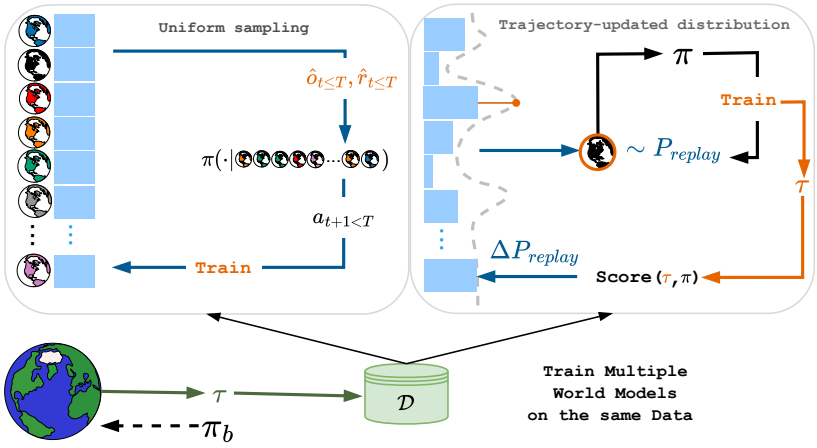


Figure 1: An overview of the two types of methods used to train on the world model ensembles

3.1 Training Multiple World Models

Solving a planning problem for every conceivable history that an agent might encounter within the CMDP is mostly intractable beyond the simplest examples (Martin, 1967; Duff, 2002; Guez et al., 2012; 2013; Zintgraf et al., 2020; Fellows et al., 2024). In this work, we assume access to a *non-sequential* offline dataset $\mathcal{D}$ of N state-action-state-reward transition observations: $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=0}^{N-1}$, all collected from a single MDP $\theta^\star$. Therefore, we address this tractability issue by learning a highly informative posterior distribution $P_\Theta(\mathcal{D})$ using offline data, which concentrates around a small region of the parameter space Θ containing the true dynamics $\theta^\star$. By doing so, we effectively reduce the hypothesis space to a manageable subset of Θ, enabling the tractable evaluation of the RL objective.

Practically, we implement this by training multiple distinct world models, each initialized with different random weights and trained on different permutations of the data. The inherent variability introduced by stochastic gradient descent during the training process causes each world model to exhibit slightly different dynamics (Amari, 1993). However, an agent trained in any one of these world models is not guaranteed to transfer well to the real environment, and it is this problem we tackle by using the ensemble of world models.

3.2 World Models as levels

If we treat each world model θ as a *level*, we can apply standard minimax regret algorithms to our setting. More formally, we consider the two-player game between an adversary G and student policy π, such that the adversary generates a level (i.e., a world model) $\theta \in \Theta$ that maximizes the agent's regret, and the agent trains as normal on the provided levels. Note, we define $\Theta \doteq \{\theta : L_2(\theta, \mathcal{D}) < \epsilon\}$ to be the set of all world models that have loss over the dataset $\mathcal{D}$ of less than some threshold ϵ. At Nash equilibrium of this game, Dennis et al. (2020) showed that the policy satisfies Equation (4). In other words, the policy's maximum regret on any $\theta \in \Theta$ is bounded by $W \doteq \min_{\pi \in \Pi}\{\max_{\theta \in \Theta}\{\text{Regret}_\theta(\pi)\}\}$. Since we have assumed that $\theta^\star \in \Theta$, *this bound further applies to the true environment dynamics*. Moreover, since the adversary is constrained to only choose levels within Θ, i.e., those that have loss less than a certain value, it cannot be overly adversarial and provide totally unrealistic dynamics to train the agent on.

In order to make this procedure practical, we use the high-performing PLR algorithm as illustrated in the right side in Figure 1, treating different world models θ as levels. Despite PLR not guaranteeing convergence to a Nash equilibrium, it generally results in improved zero-shot generalisation to out-of-distribution tasks. Since regret for a given world model is not always known, we use the standard regret approximations of Positive Value Loss for level θ_i:

$$S_i = \frac{1}{T}\sum_{t=0}^{T} \max\left(\sum_{k=t}^{T}(\gamma\lambda)^{k-t}\delta_k, 0\right). \tag{5}$$

To ensure that the agent does not overfit to the training distribution of learned world models, we implement the algorithm illustrated on the left side of Figure 1 where the agent experiences state transitions from multiple world models within *one single* episode.

4 Experimental Setup

This section outlines our experimental setup, covering offline data collection, world model training, RL process details as well as baseline descriptions.

4.1 Dataset Curation

One key principle guiding our dataset curation strategy is the concept of state coverage. Relying on a single behavior policy π_b often results in exploration of only a limited subset of the entire state space. To address this limitation, we employ multiple behavior policies to gather diverse data. Specifically, we train an agent in the real environment using Proximal Policy Optimization (PPO) (Schulman et al., 2017) and periodically create checkpoints throughout the training process. These checkpoints serve as distinct behavior policies, ensuring that our dataset encompasses a wide range of behaviors—from those generated by randomly initialized policies to those that effectively solve the task. This ensures that the world models fits a sufficient state support for the agent to plan and explore. This approach is not informed by any algorithmic insight like pessimism (Kumar et al., 2020) or behavior cloning regularization (Fujimoto & Gu, 2021). We note that our dataset is shuffled in the level of state transitions and *does not* require sequences to train the world models.

The frequency of checkpointing and the number of trajectories collected at each checkpoint are determined heuristically to optimize collection and world model training time, taking into account the environment's episode length and the performance metrics. Figure 2 demonstrates the schedule for collecting behavior policy trajectories in the Hopper environment, illustrating how different stages of the agent's training are captured to reflect varying levels of proficiency.

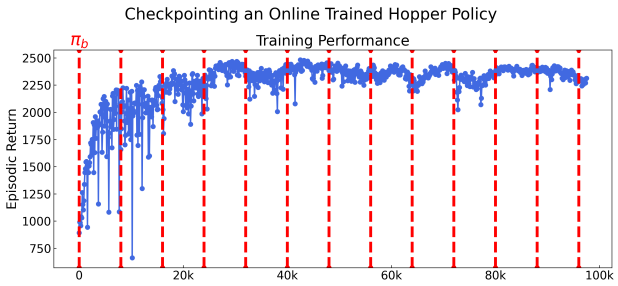


Figure 2: Data collection in the real environment using different π_b checkpoints marked by the vertical lines are used to collect trajectories for $\mathcal{D}$

4.2 World Model Training

In order to test our method for different numbers of collected transitions, we uniformly subsample the transitions dataset $\mathcal{D}$ to varying sizes. The world models are trained on the same data with an L_2 test loss but show different final test losses and slightly different dynamics. We train every model to convergence. The world models in our experiments are implemented as fully connected networks with a concatenated input of actions and observations and an output of the concatenated next observations and reward. We then use different seeds to initialize the world model network, and shuffle the order the data is passed through it. More training details and results can be found on Table 3 in Appendix A.2.

At inference time, we add Gaussian Noise to the outputs equal to the square root of the L_2 test loss, to represent the fact that the world models are modelling a distribution over next states.

4.3 Training the Reinforcement Learning Agent

The agent is implemented as a recurrent actor-critic network following the template from PureJaxRL (Lu et al., 2022). The agent's actions depend on the current observation and episode trajectory, implemented as the recurrent state of the actor-critic network. We use the recurrent state to test the agent's ability to perform system identification across the world models it is trained on. This is also done to verify that the world models have distinct dynamics. Visualizations and analysis of the hidden state can be found in A.4.

We implement a suite of methods to train using the world models ensemble:

PLR: Prioritized Level Replay as described in with an L_1 value loss score function,

PLR_PVL: PLR with a Positive Value Loss scoring as described in 3.2 to approximate regret,

DR: Domain Randomization implemented by randomly selecting a new world model θ from a uniform distribution over the world model buffer Θ_{train}

DR_STEP where we change θ_i at every individual timestep of the agent in a fixed length episode instead of only doing it at every reset.

DR_PROB where we change θ_i at every individual timestep with probability p which is sampled every step. The probability p could also serve as a classic UED parameter where it is varied based on the episode's score.

WM: *a single world model* without any adversity or curriculum curation as a baseline for the aforementioned methods.

To address policy overfitting to the world models' dynamics without querying the real environment, we use holdout world models trained on transitions from the validation set that is also used to validate the supervised world model training. We observe that when overfitting occurs, as indicated by the decoupling of training and evaluation episodic rewards, the standard deviation of the policy's rollout episodic rewards across the holdout world models increases significantly. This phenomenon serves as a reliable indicator for early stopping and helps prevent policy overfitting. We note that our method and hyperparameters do not rely on online tuning.

The PLR implementations are based on JaxUED (Coward et al., 2024). We evaluate the policies on the same environment the world model training dataset was collected in and use RLiable library as presented by Agarwal et al. (2021) to measure the performance. Every metric is plotted within a shaded area the marking 95% confidence interval calculated over five seeds and 50 episodes on each respective environment. The entire pipeline, from data collection to wold model and subsequent policy training is implemented in the JAX Ecosystem (DeepMind et al., 2020) to leverage hardware acceleration and speed up training.

4.4 Baselines

We baseline our methods by training on a randomly sampled single world model (**WM**) and against commonplace offline RL algorithms like CQL (Kumar et al., 2020) and SAC$_N$ (An et al., 2021). We implement these algorithms ourselves and verify that they achieve the documented performance on the D4RL dataset (Fu et al., 2020), the benchmarks for which they were originally developed and evaluated. To ensure a fair comparison, we structure our collected trajectories to match the format and characteristics of the datasets the algorithms were designed for, utilizing the same trajectories *without* shuffling them at the transition level. Each of the baselines is tuned by doing a grid search of the ranges documented in their respective papers. The specific ranges can be found on Table 6 and Table 7.

5 Results

In this section we show the most notable results that elucidate important aspect of our approach. A complete compilation of the results can be found in the Appendix. We collect data from and evaluate on environments from the Gymnax (Lange, 2022) and Brax (Freeman et al., 2021) suites. All the evaluations are performed on full trajectories across five random seeds on the corresponding real environments.

5.1 Preventing Exploitation

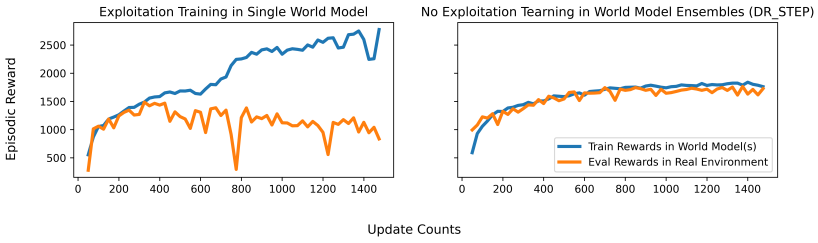


Figure 3: Preventing reward hijacking of the learned model by using the ensemble training method

Training in world model ensembles prevents the agents from overfitting to the training distribution and hacking the rewards. Figure 3 shows the results on a world models trained with $2{\cdot}10^4$ transitions, only 20 episodes worth of transitions.

5.2 Classic Control

The suite of methods using world model ensembles outperforms naive world model training with only a couple of episodes worth of transitions from dataset $\mathcal{D}$. We illustrate the evaluation on the Cartpole environment in Figure 6 to showcase the effectiveness of world model ensembles to reach the highest episodic return possible in less than half the transition counts compared to using a single world model. Training on multiple world models beats the single world models baseline in a simple environment. Figure 5 shows our methods consistently outperform training on a single world model for sparser data and even achieve returns higher than the behavior policy that was learned online.

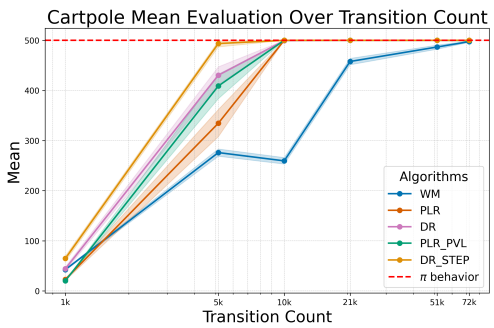


Figure 4: Mean of the evaluations on Cartpole

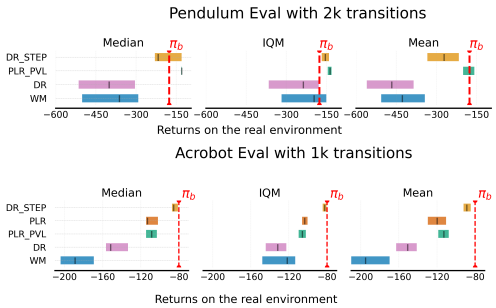


Figure 5: Interquartile Mean (IQM), Mean, and Median of the world model ensemble trained policy evaluated on the real environment

5.3 Robotic Locomotion in Brax

We test our model on Hopper and HalfCheetah from the Brax suite of environment. We notice that the methods that sample a new level uniformly at every step or with a probability p outperform every method in sparser data regimes.

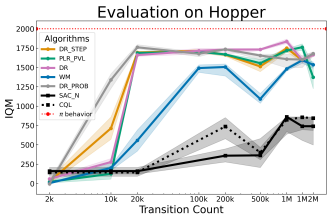


Figure 6: IQM for Hopper

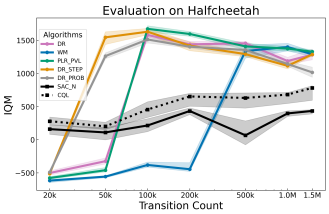


Figure 7: IQM for Half Cheetah

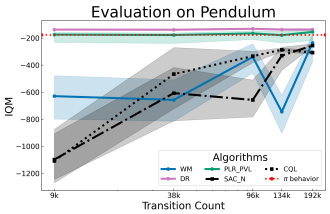


Figure 8: IQM for Pendulum

5.4 RNN Analysis

Table 1: Classification accuracy of 9 world models and the real environment

% of $\lvert\mathcal{D}\rvert$	DR	PLR	PLR_PVL
1	0.68	0.11	0.47
5	0.41	0.65	0.67
10	0.62	0.68	0.40
20	0.67	0.67	0.09
50	0.76	0.66	0.36
70	0.68	0.58	0.37
100	0.54	0.85	0.79

Our claim is that the world models have sufficiently distinct dynamics and can therefore serve as different contextual MDPs. If true, regret-based training should help the agent adapt to all these dynamics. We demonstrate this by deploying our agent across multiple world models and on the real environment. We then train a classifier on the recurrent states of said agent to identify its environment and achieve an average of 62% accuracy on the **DR**, 60% on **PLR** and 45% on **PLR_PVL**; all above the 10% random prediction accuracy. More in A.5,A.4.

6 Related Work

Reinforcement Learning has achieved impressive results, some of the most notable ones being Go (Silver et al., 2016b), Starcraft (Vinyals et al., 2019), Atari (Mnih et al., 2015) and more recent advances focusing on multi-task generalizations (Bruce et al., 2024; Hafner et al., 2023). Despite these impressive results, RL methods fail to generalize to settings even slightly different than the training environments (Cobbe et al., 2019; Mediratta et al., 2023), indicating that the generalization to real world settings remains an open challenge.

The generalization of an RL agent can be enhanced by ensuring it is exposed to a sufficiently diverse set of environments in training time. The Unsupervised Environment Design (UED) (Dennis et al., 2020; Jiang et al., 2021a) line of work achieves this by relaxing the definition of the environment to a combinatorially large set of possible configurations captured by a set of parameters, commonly referred to as *levels*. The choice of the parameter space is specifically tailored to the general task domain also known as the underspecified environments (e.g. a maze environment is parameterized by the placement of the walls, start and goal position whereas a one dimensional bipedal environment is parameterized by the roughness of the terrain). UED uses Minimax regret (Savage, 1951) to make the agent robust to the most challenging environment configurations without prior knowledge of which set of parameters it will act in. While these approaches are meant to exemplify deployment in challenging situations, they remain reliant on semantically informed choices of parameters that capture useful *levels* of difficulty (the color of the background is not as useful in curating the training of a bipedal walker as the roughness of the train). Jiang et al. (2021a) was very helpful in bridging the intuitive algorithm of Prioritized Level Replay with new regret approximations that provide minimax guarantees.

World models (Ha & Schmidhuber, 2018) propose a different approach where the agent is equipped with a compact representation of the real environments trained using a dataset of transitions in said environment. More recent work shows that world models can serve as task-agnostic Continual Reinforcement Learning baselines (Kessler et al., 2023) or used in online RL to achieve human-level performance on Atari (Hafner et al., 2020). In principles, world modelling does not hinge on task-specific heuristics and only relies on increasing the robustness of the agent by tuning the uncertainty inside the world model.

A recent combination of the world model and *Minimax Regret* approach by Rigter et al. (2023) trains a world model that can derive robust policies. This is done through an exploration policy seeking maximal model uncertainty, similar to the self-supervised world model methods by Sekar et al. (2020). These are ultimately online methods and require sufficient exploration of states that can be physically dangerous to the agent and disrupt operation altogether (Kumar et al., 2020; 2021).

Offline RL work has provided a useful signal on the importance of using offline datasets (Kumar et al., 2020; 2021), the common challenges that arise form the distribution shift between the behavior and learned policy (Levine et al., 2020) and model error (Saleh et al., 2022) alongside the most common workarounds like truncated rollouts (Jackson et al., 2024). Model-based offline (Rigter et al., 2022) and online (Chua et al., 2018) RL methods have served as useful blueprints to manage uncertainty through *multiple* dynamic models.

The work of Li & Liang (2018) and the foundational work of Amari (1993) have paved the intuition that shuffling the data and most importantly, changing the initializations, would be effective in training sufficiently distinct models on a shared offline dataset.

7 Discussion

7.1 Dataset distributions

While our method achieves competitive results in world models trained on our dataset with wide state coverage, we do not match the same full rollout performance in D4RL datasets. We present the following investigation into why that is the case and why we think this points out to inherent biases in the field of offline RL that stand in the way of making use of data on the larger scale, similar to the fields mentioned in the introduction of this work.

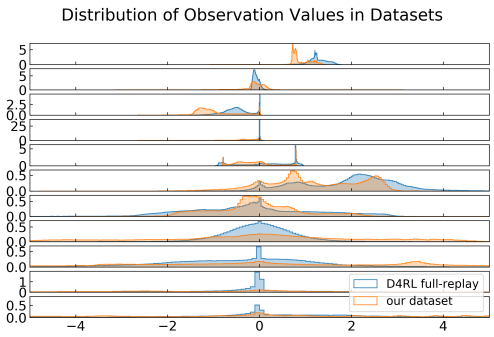


Figure 9: Observation Distribution in Hopper-full-replay datasets from D4RL and in ours

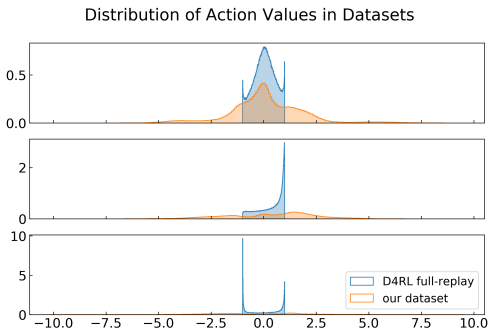


Figure 10: Action Distribution in Hopper-full-replay datasets and in ours

Previous work by Li et al. (2024) has shown that offline RL methods are susceptible to implicit biases in the data collection practice. Figure 9 offers a succinct qualitative analysis by showing that more than half of the Hopper dimensions from D4RL have narrower coverage and bias the agent towards healthy behavior; a helpful addition for hopper as the unhealthy state flag can cause an early termination and vastly affect evaluation. A method that includes a Behavior Cloning term like TD3+BC (Fujimoto & Gu, 2021) is at a clear advantage since it is directly biased away from unhealthy states that would otherwise be explored more in the online environment (as our dataset distribution shows in Figures 9 through 11. The state of offline RL and its benchmarks has *positively reinforced* a direction of methods that does not account for the type increasingly available large scale datasets.

Moreover, this work would benefit from a more principled and interpretable method of sampling the possible world models from Θ set – as defined in 3.2 – other than simply changing the shuffling and initialization seeds. A natural extension is that of level generation to have an expanding buffer of available levels during the adversarial training. Our method also offers a way to generate an RL training curricula by abstracting away hand-crafted heuristics and using data to generate different levels directly. Such tools should not be exclusive to offline RL.

Finally, the results in physical engines like Brax should be extended to *real* physical platforms and address the engineering challenges posed by the *sim2real* gap, especially in sensitive settings where online training can be physically hazardous.

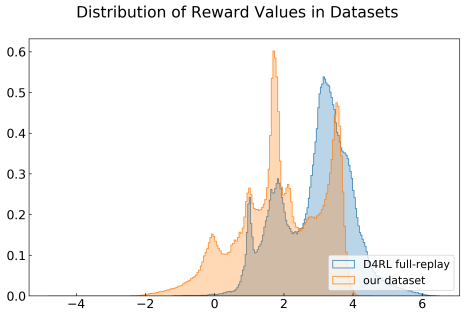


Figure 11: Reward Distribution in Hopper-full-replay datasets and in ours

8 Conclusion

In this work we present a novel way to guarantee transfer robustness to the real environment over world models fitted on offline data. To the best of our knowledge, this is the first work that performs adversarial training under this specific fully parametric constraint. Our method naturally lends itself to other architectures and hopefully will help blaze the trails towards meaningful deployment of state-of-the-art RL algorithms into the *real* world based on training inside large scale generative models.

REFERENCES

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

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5 (4-5):185–196, 1993.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

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.

Michael Beukman, Samuel Coward, Michael T. Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob N. Foerster. Refining minimax regret for unsupervised environment design. *CoRR*, abs/2402.12284, 2024. doi: 10.48550/ARXIV.2402.12284. URL https://doi.org/10.48550/arXiv.2402.12284.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Thomas Kleine Buening, Christos Dimitrakakis, Hannes Eriksson, Divya Grover, and Emilio Jorge. Minimax-bayes reinforcement learning. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 7511–7527. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/buening23a.html.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Samuel Coward, Michael Beukman, and Jakob Foerster. Jaxued: A simple and useable ued library in jax. *arXiv preprint arXiv:2403.13091*, 2024.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

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. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Michael O'Gordon Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, 2002. AAI3039353.

Mattie Fellows, Brandon Kaplowitz, Christian Schroeder de Witt, and Shimon Whiteson. Bayesian exploration networks. In *ICML*, 2024.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/35051070e572e47d2c26c241ab88307f-Paper.pdf.

Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *Journal of Artificial Intelligence Research*, 48:841–883, 10 2013. doi: 10.1613/jair.4117.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

Imagen-Team-Google, Jason Baldridge, Jakob Bauer, Mukul Bhutani, Nicole Brichtova, Andrew Bunner, Kelvin Chan, Yichang Chen, Sander Dieleman, Yuqing Du, Zach Eaton-Rosen, Hongliang Fei, Nando de Freitas, Yilin Gao, Evgeny Gladchenko, Sergio Gómez Colmenarejo, Mandy Guo, Alex Haig, Will Hawkins, Hexiang Hu, Huilian Huang, Tobenna Peter Igwe, Christos Kaplanis, Siavash Khodadadeh, Yelin Kim, Ksenia Konyushkova, Karol Langner, Eric Lau, Shixin Luo, Soňa Mokrá, Henna Nandwani, Yasumasa Onoe, Aäron van den Oord, Zarana Parekh, Jordi Pont-Tuset, Hang Qi, Rui Qian, Deepak Ramachandran, Poorva Rane, Abdullah Rashwan, Ali Razavi, Robert Riachi, Hansa Srinivasan, Srivatsan Srinivasan, Robin Strudel, Benigno Uria, Oliver Wang, Su Wang, Austin Waters, Chris Wolff, Auriel Wright, Zhisheng Xiao, Hao Xiong, Keyang Xu, Marc van Zee, Junlin Zhang, Katie Zhang, Wenlei Zhou, Konrad Zolna, Ola Aboubakar, Canfer Akbulut, Oscar Akerlund, Isabela Albuquerque, Nina Anderson, Marco Andreetto, Lora Aroyo, Ben Bariach, David Barker, Sherry Ben, Dana Berman, Courtney Biles, Irina Blok, Pankil Botadra, Jenny Brennan, Karla Brown, John Buckley, Rudy Bunel, Elie Bursztein, Christina Butterfield, Ben Caine, Viral Carpenter, Norman Casagrande, Ming-Wei Chang, Solomon Chang, Shamik Chaudhuri, Tony Chen, John Choi, Dmitry Churbanau, Nathan Clement, Matan Cohen, Forrester Cole, Mikhail Dektiarev, Vincent Du, Praneet Dutta, Tom Eccles, Ndidi Elue, Ashley Feden, Shlomi Fruchter, Frankie Garcia, Roopal Garg, Weina Ge, Ahmed Ghazy, Bryant Gipson, Andrew Goodman, Dawid Górny, Sven Gowal, Khyatti Gupta, Yoni Halpern, Yena Han, Susan Hao, Jamie Hayes, Amir Hertz, Ed Hirst, Tingbo Hou, Heidi Howard, Mohamed Ibrahim, Dirichi Ike-Njoku, Joana Iljazi, Vlad Ionescu, William Isaac, Reena Jana, Gemma Jennings, Donovon Jenson, Xuhui Jia, Kerry Jones, Xiaoen Ju, Ivana Kajic, Christos Kaplanis, Burcu Karagol Ayan, Jacob Kelly, Suraj Kothawade, Christina Kouridi, Ira Ktena, Jolanda Kumakaw, Dana Kurniawan, Dmitry Lagun, Lily Lavitas, Jason Lee, Tao Li, Marco Liang, Maggie Li-Calis, Yuchi Liu, Javier Lopez Alberca, Peggy Lu, Kristian Lum, Yukun Ma, Chase Malik, John Mellor, Inbar Mosseri, Tom Murray, Aida Nematzadeh, Paul Nicholas, João Gabriel Oliveira, Guillermo Ortiz-Jimenez, Michela Paganini, Tom Le Paine, Roni Paiss, Alicia Parrish, Anne Peckham, Vikas Peswani, Igor Petrovski, Tobias Pfaff, Alex Pirozhenko, Ryan Poplin, Utsav Prabhu, Yuan Qi, Matthew Rahtz, Cyrus Rashtchian, Charvi Rastogi, Amit Raul, Ali Razavi, Sylvestre-Alvise Rebuffi, Susanna Ricco, Felix Riedel, Dirk Robinson, Pankaj Rohatgi, Bill Rosgen, Sarah Rumbley, Moonkyung Ryu, Anthony Salgado,

Sahil Singla, Florian Schroff, Candice Schumann, Tanmay Shah, Brendan Shillingford, Kaushik Shivakumar, Dennis Shtatnov, Zach Singer, Evgeny Sluzhaev, Valerii Sokolov, Thibault Sottiaux, Florian Stimberg, Brad Stone, David Stutz, Yu-Chuan Su, Eric Tabellion, Shuai Tang, David Tao, Kurt Thomas, Gregory Thornton, Andeep Toor, Cristian Udrescu, Aayush Upadhyay, Cristina Vasconcelos, Alex Vasiloff, Andrey Voynov, Amanda Walker, Luyu Wang, Miaosen Wang, Simon Wang, Stanley Wang, Qifei Wang, Yuxiao Wang, Ágoston Weisz, Olivia Wiles, Chenxia Wu, Xingyu Federico Xu, Andrew Xue, Jianbo Yang, Luo Yu, Mete Yurtoglu, Ali Zand, Han Zhang, Jiageng Zhang, Catherine Zhao, Adilet Zhaxybay, Miao Zhou, Shengqi Zhu, Zhenkai Zhu, Dawn Bloxwich, Mahyar Bordbar, Luis C. Cobo, Eli Collins, Shengyang Dai, Tulsee Doshi, Anca Dragan, Douglas Eck, Demis Hassabis, Sissie Hsiao, Tom Hume, Koray Kavukcuoglu, Helen King, Jack Krawczyk, Yeqing Li, Kathy Meier-Hellstern, Andras Orban, Yury Pinsky, Amar Subramanya, Oriol Vinyals, Ting Yu, and Yori Zwols. Imagen 3, August 2024.

Matthew Thomas Jackson, Michael Tryfan Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and Jakob Foerster. Policy-guided diffusion. *arXiv preprint arXiv:2404.06356*, 2024.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021b.

Samuel Kessler, Mateusz Ostaszewski, MichałPaweł Bortkiewicz, Mateusz Żarski, Maciej Wolczyk, Jack Parker-Holder, Stephen J Roberts, Piotr Mi, et al. The effectiveness of world models for continual reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 184–204. PMLR, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020.

Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Anqi Li, Dipendra Misra, Andrey Kolobov, and Ching-An Cheng. Survival instinct in offline reinforcement learning. *Advances in neural information processing systems*, 36, 2024.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31, 2018.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

J. J. Martin. *Bayesian decision problems and Markov chains [by] J. J. Martin*. Wiley New York, 1967.

Ishita Mediratta, Qingfei You, Minqi Jiang, and Roberta Raileanu. The generalization gap in offline reinforcement learning. *arXiv preprint arXiv:2312.05742*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928. URL http://eudml.org/doc/159291.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

Marc Rigter, Minqi Jiang, and Ingmar Posner. Reward-free curricula for training robust world models. *arXiv preprint arXiv:2306.09205*, 2023.

Esra' Saleh, John D Martin, Anna Koop, Arash Pourzarabi, and Michael Bowling. Should models be accurate? *arXiv preprint arXiv:2205.10736*, 2022.

Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pp. 8583–8592. PMLR, 2020.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016a. ISSN 1476-4687. doi: 10.1038/nature16961.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016b.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0-262-03924-9.

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, February 2023.

Georgios Tzannetos, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum with task correlations for deep reinforcement learning. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 5027–5036. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/556. URL https://doi.org/10.24963/ijcai.2024/556. Main Track.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Abraham Wald. An Essentially Complete Class of Admissible Decision Functions. *The Annals of Mathematical Statistics*, 18(4):549 – 555, 1947. doi: 10.1214/aoms/1177730345. URL https://doi.org/10.1214/aoms/1177730345.

Abraham Wald. *Statistical decision functions*. Wiley publications in statistics. John Wiley & Sons, New York : London, 1950.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/pdf?id=Hkl9JlBYvr.

A Appendix

A.1 UED Discussion

In this section we revisit the main principles of UED and their connection to Bayesian RL. Our derivation reveals that minimax UED is equivalent to learning a Bayes-optimal policy under a least favourable prior. As Bayesian RL is a more general framework that allows for optimality under different priors, we now discuss the relative advantages and disadvantages of choosing a least favourable prior. The benefits of choosing a least favourable prior include:

I. Policies are robust to changes in prior A key advantage of the least favourable prior is that policies can be robust to changes in belief. When the minimax theorem (Neumann, 1928) holds, a Nash equilibrium to the two-player game exists with solution $(\pi_{\text{MinMax}}, \Theta_{\text{max}}^{\pi_{\text{MinMax}}})$ and it follows (Buening et al., 2023):

$$\min_{\pi \in \Pi_{\mathcal{H}}} \max_{\theta \in \Theta} [\text{Regret}_\theta(\pi)] = \min_{\pi \in \Pi_{\mathcal{H}}} \max_{P \in \mathcal{P}} \mathbb{E}_{\theta \sim P} [\text{Regret}_\theta(\pi)] = \max_{P \in \mathcal{P}} \min_{\pi \in \Pi_{\mathcal{H}}} \mathbb{E}_{\theta \sim P} [\text{Regret}_\theta(\pi)], \quad (6)$$

which implies that the minimax policy is robust to any change in the prior.

II. Protection against worst case MDPs The set $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$ indexes MDPs where policies have the worst possible regret. This ensures that the agent following π_{MinMax} at test time is protected against situations where the return has the potential to be very low. From a safety perspective, this can protect an agent from behaving in a way that is dangerous towards itself or others in an environment; in particular, if an agent is at a Nash equilibrium, the regret across all MDPs is bounded by $\min_{\pi \in \Pi_{\mathcal{H}}} \max_{\theta \in \Theta} [\text{Regret}_\theta(\pi)]$.

There are also several drawbacks to choosing a least favourable prior. Many of these stem from the restriction of the prior to $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$, and include:

I. Inability to exploit prior knowledge The least favourable prior excludes the ability to integrate pre-existing beliefs into the Bayes-optimal policy. If prior knowledge about the set of environments is available, for example from and offline dataset or known skills that are common across all environments, this information cannot be exploited by a least favourable prior. This is most pertinent if the true distribution over context variables is known a priori, as using this as the prior results in the greatest regret reduction according to the frequency in which MDPs are encountered in practice.

II. Inability to learn optimal policies For proper priors with support over Θ, provided $\theta^\star \in \Theta$, a key property of Bayes-optimal policies is that they tend towards the optimal policy $\pi(s_t, \theta^\star)$ in the limit of $t \to \infty$. If the index $\theta^\star$ of true MDP allocated to the agent at test time lies outside of the set of worst regret parameters, that is $\theta^\star \notin \Theta_{\text{max}}^{\pi_{\text{MinMax}}}$, then the posterior under the least favourable prior cannot collapse to place its support on $\theta^\star$ and the corresponding policy will never be optimal for $\mathcal{M}(\theta^\star)$. As $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$ is typically a very small subset of Θ and the whole of $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$ is never learned in practice, we expect this situation to be frequently encountered. This point has been observed empirically as the inability to generalise to out of distribution tasks (Jiang et al., 2021a).

III. Issues with learning Nash equilibria The conditions needed to prove the existence of the minimax solution - a finite state-action space, a finite horizon, known reward, a finite set of MPDs (see Buening et al. (2023) for details) - rarely hold in a CMDP in practice. Whilst it is currently unknown whether the minimax theorem can be generalised to more realistic CMDPs, empirical evidence suggests this is not the case (Buening et al., 2023). MDPs where the Nash equilibrium does not exist present a convergence issue when learning a minimax policy. Moreover, even if the Nash equilibrium exists, algorithms rarely learn the entirety of $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$ required for the minimax policy (Beukman et al., 2024). In particular, if the algorithm collapses to a prior with support over single context variable, we cannot expect the minimax policy to learn anything useful at test time.

IV. Inherent pessimism A least favourable prior encodes the most pessimistic belief possible - that an agent will always be faced with a set of MDPs that have the potential for the highest regret. The agent does not consider any hypothesis outside of $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$ when reasoning about its beliefs, despite the fact these MPDs may be more typical of the environments encountered at test time. This

prevents exploration of alternative hypotheses and is not a universally appropriate belief for every CMDP.

V. Loss of admissibility A key benefit of Bayes-optimal policies is that, given a proper prior, they are guaranteed to be admissible - they cannot be Pareto improved upon in terms of expected return $J^{\pi}(\theta)$ across Θ (Wald, 1947; 1950). Least favourable priors are not guarenteed to be proper and there exist known counterexamples where inadmissible decisions are taken under a minimax policy.

VI. Amplifying effects of model misspecification In most learning settings, it is not reasonable to assume that the practitioner can specify a CMPD that contains the exact space of MDPs that an agent could encounter. We must account for some degree of misspecification where there exist subsets of context variables $\Theta' \subset \Theta$ that do not correspond to a realisable model. By restricting the prior to have support over $\Theta_{\text{max}}^{\pi_{\text{MinMax}}}$, it may occur that the prior only has support over MDPs in Θ', hence the corresponding minimax policy will only account for MDPs that do not exist in practice.

Like any prior, we see that choice of using a least favourable prior is *subjective*, and its justification depends on weighing up the relative advantages and disadvantages by a practitioner on a case-by-case basis. Either way, the least favourable prior and minimax solution is by no means a universally appropriate method.

A.2 World Model Training Results

Table 2: Transition Counts for each dataset

Environment	Transition Count
Acrobot	$1.02 \cdot 10^5$
Cartpole	$1.02 \cdot 10^5$
Mountaincar	$1.03 \cdot 10^5$
Pendulum	$1.92 \cdot 10^5$
Hopper	$2 \cdot 10^6$
HalfCheetah	$2 \cdot 10^6$

A.3 Hyperparameters

A.4 Hidden States Visualization

The PCA for RNN states of different agents trained with different algorithms

Each row illustrates the episodic progression, with Figure 12 depicting the 2-dimensional Principal Component Analysis (PCA) of the 256-dimensional hidden states. These hidden states are collected from 10 differently initialized rollouts of the same agent. The rollouts are performed across 9 different world models and the real environment, ensuring a fair and balanced classification dataset. Notably, no pattern of stability emerges with the **DR**-trained agent. However, the **PLR** and **PLR_PVL** agents exhibit stabilization midway through the episode, within a smaller range on the principal components compared to the PCA of their initial state. While this warrants further investigation, we can intuitively infer that the agent learns to act optimally across all world models, and that this optimal behavior tends to become increasingly similar—**though still distinct**—across the different world models and environments.

A.5 Hidden States Classification

The confusion matrix for the classification of the world model using the agent's recurrent state from all the steps of the episode.

Table 3: L_2 loss in world model training results for different $\mathcal{D}$ ratios across environment

Environment	% of $\lvert\mathcal{D}\rvert$	Train Loss Mean	Train Loss Median	Test Loss Mean	Test Loss Median
Pendulum-v1	1	$1.201 \cdot 10^{-7}$	$1.19 \cdot 10^{-7}$	$5.87 \cdot 10^{-4}$	$5.83 \cdot 10^{-4}$
	5	$2.20 \cdot 10^{-6}$	$2.19 \cdot 10^{-6}$	$5.93 \cdot 10^{-5}$	$5.91 \cdot 10^{-5}$
	10	$4.28 \cdot 10^{-6}$	$4.39 \cdot 10^{-6}$	$3.02 \cdot 10^{-5}$	$3.01 \cdot 10^{-5}$
	20	$6.85 \cdot 10^{-6}$	$6.90 \cdot 10^{-6}$	$1.87 \cdot 10^{-5}$	$1.86 \cdot 10^{-5}$
	50	$9.35 \cdot 10^{-6}$	$9.34 \cdot 10^{-6}$	$1.33 \cdot 10^{-5}$	$1.34 \cdot 10^{-5}$
	70	$3.99 \cdot 10^{-1}$	$1.02 \cdot 10^{-5}$	$4.08 \cdot 10^{-1}$	$1.28 \cdot 10^{-5}$
	100	$3.99 \cdot 10^{-1}$	$1.11 \cdot 10^{-5}$	$4.08 \cdot 10^{-1}$	$1.23 \cdot 10^{-5}$
Acrobot	1	$8.86 \cdot 10^{-7}$	$9.11 \cdot 10^{-7}$	$1.20 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$
	5	$7.53 \cdot 10^{-6}$	$7.35 \cdot 10^{-6}$	$2.55 \cdot 10^{-3}$	$2.57 \cdot 10^{-3}$
	10	$1.71 \cdot 10^{-5}$	$1.69 \cdot 10^{-5}$	$1.17 \cdot 10^{-3}$	$1.18 \cdot 10^{-3}$
	20	$3.37 \cdot 10^{-5}$	$3.37 \cdot 10^{-5}$	$5.05 \cdot 10^{-4}$	$5.05 \cdot 10^{-4}$
	50	$7.60 \cdot 10^{-5}$	$7.60 \cdot 10^{-5}$	$3.01 \cdot 10^{-4}$	$3.02 \cdot 10^{-4}$
	70	$9.14 \cdot 10^{-5}$	$9.09 \cdot 10^{-5}$	$2.67 \cdot 10^{-4}$	$2.66 \cdot 10^{-4}$
	100	$1.40 \cdot 10^{-4}$	$1.39 \cdot 10^{-4}$	$2.81 \cdot 10^{-4}$	$2.81 \cdot 10^{-4}$
Cartpole	1	$1.95 \cdot 10^{-8}$	$1.86 \cdot 10^{-8}$	$3.57 \cdot 10^{-5}$	$3.60 \cdot 10^{-5}$
	5	$2.97 \cdot 10^{-7}$	$2.89 \cdot 10^{-7}$	$4.20 \cdot 10^{-6}$	$4.15 \cdot 10^{-6}$
	10	$4.86 \cdot 10^{-7}$	$4.85 \cdot 10^{-7}$	$2.22 \cdot 10^{-6}$	$2.23 \cdot 10^{-6}$
	20	$6.49 \cdot 10^{-7}$	$6.47 \cdot 10^{-7}$	$1.52 \cdot 10^{-6}$	$1.52 \cdot 10^{-6}$
	50	$8.05 \cdot 10^{-7}$	$8.03 \cdot 10^{-7}$	$1.15 \cdot 10^{-6}$	$1.14 \cdot 10^{-6}$
	70	$8.61 \cdot 10^{-7}$	$8.61 \cdot 10^{-7}$	$1.08 \cdot 10^{-6}$	$1.08 \cdot 10^{-6}$
	100	$8.98 \cdot 10^{-7}$	$8.98 \cdot 10^{-7}$	$1.05 \cdot 10^{-6}$	$1.04 \cdot 10^{-6}$
Hopper	0.1	$3.42 \cdot 10^{-3}$	$3.41 \cdot 10^{-3}$	$1.51 \cdot 10^{-2}$	$1.51 \cdot 10^{-2}$
	0.5	$2.71 \cdot 10^{-3}$	$2.58 \cdot 10^{-3}$	$1.19 \cdot 10^{-2}$	$1.19 \cdot 10^{-2}$
	1.0	$1.88 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	$1.04 \cdot 10^{-2}$	$8.79 \cdot 10^{-3}$
	5.0	$1.47 \cdot 10^{-3}$	$1.01 \cdot 10^{-3}$	$9.09 \cdot 10^{-3}$	$8.05 \cdot 10^{-3}$
	10.0	$1.21 \cdot 10^{-3}$	$2.30 \cdot 10^{-4}$	$8.15 \cdot 10^{-3}$	$7.40 \cdot 10^{-3}$
	25.0	$1.08 \cdot 10^{-3}$	$3.21 \cdot 10^{-4}$	$7.41 \cdot 10^{-3}$	$6.24 \cdot 10^{-3}$
	50.0	$9.71 \cdot 10^{-4}$	$3.32 \cdot 10^{-4}$	$6.82 \cdot 10^{-3}$	$5.10 \cdot 10^{-3}$
	75.0	$8.87 \cdot 10^{-4}$	$3.16 \cdot 10^{-4}$	$6.31 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$
	100.0	$8.20 \cdot 10^{-4}$	$3.02 \cdot 10^{-4}$	$5.91 \cdot 10^{-3}$	$4.36 \cdot 10^{-3}$
HalfCheetah	0.1	$8.9 \cdot 10^{-3}$	$8.8 \cdot 10^{-3}$	$3.6 \cdot 10^{-2}$	$3.6 \cdot 10^{-2}$
	0.5	$6.3 \cdot 10^{-3}$	$5.9 \cdot 10^{-3}$	$2.8 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$
	1.0	$4.3 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$	$2.3 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$
	5.0	$3.4 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
	10.0	$2.8 \cdot 10^{-3}$	$5.6 \cdot 10^{-4}$	$1.6 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
	25.0	$2.4 \cdot 10^{-3}$	$5.2 \cdot 10^{-4}$	$1.3 \cdot 10^{-2}$	$9.2 \cdot 10^{-3}$
	50.0	$2.1 \cdot 10^{-3}$	$4.9 \cdot 10^{-4}$	$1.2 \cdot 10^{-2}$	$5.5 \cdot 10^{-3}$
	75.0	$1.9 \cdot 10^{-3}$	$4.7 \cdot 10^{-4}$	$1.1 \cdot 10^{-2}$	$4.6 \cdot 10^{-3}$
	100.0	$1.7 \cdot 10^{-3}$	$4.2 \cdot 10^{-4}$	$9.5 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$

Table 4: Hyperparameters for the world model training

Hyperparameter	Value
Learning Rate	$1 \cdot 10^{-4}$
Batch Size	64
Hidden Size	256
Epochs	400

Table 5: Hyperparameters for Each RL Environment

Hyperparameter	**Acrobot**	**CartPole**	**Hopper**	**HalfCheetah**	**Pendulum**
Learning Rate	$5 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Number of Environments	16	4	512	16	32
Total Timesteps	$5 \cdot 10^{5}$	$5 \cdot 10^{5}$	$5 \cdot 10^{7}$	$5 \cdot 10^{7}$	$1 \cdot 10^{7}$
PPO Update Epochs	4	4	4	64	4
Number of Minibatches	4	4	32	4	4
Gamma	0.99	0.99	0.99	0.99	0.99
GAE Lambda	0.95	0.95	0.95	0.95	0.95
Clip EPS	0.2	0.2	0.2	0.2	0.2
Entropy Coefficient	0.01	0.01	0.0	0.003	0.01
Value Function Coef	0.5	0.5	0.5	0.5	0.5
Max Grad Norm	1	0.5	0.5	1	1.0
Activation Function	tanh	tanh	tanh	tanh	tanh
Anneal Learning Rate	true	true	false	true	true
Number of Eval Envs	1	1	1	1	1
Eval Frequency	4	4	100	4	4

Table 6: Hyperparameter range sweep for SAC N

Hyperparameter	**Values**
polyak step size	[0.004, 0.006]
gamma	0.99, 0.999
lr	5×10^{-5}, 1×10^{-4}, 2×10^{-4}, 3×10^{-4}
num of critics	200, 300, 500
batch size	128, 256, 512

Table 7: Hyperparameter range sweep for CQL

Hyperparameter	**Values**
polyak step size	[0.004, 0.006]
gamma	0.99, 0.999
lr	5×10^{-5}, 1×10^{-4}, 3×10^{-4}
num critics	200, 300, 500
batch size	128, 256, 512
seed	1, 2, 3
cql target actions gap	[0.5, 2.0]
cql temperature	[0.5, 2.0]
cql min q weight	[1.0, 10.0]
cql n actions	5, 10, 15

Table 8: Comparison of Methods and 100-step Rewards

Method	**100-step Reward**
Behavior Cloning	222.23
Randomly Initialized Policy	97.47
DR-step	148.17
DR	149.20

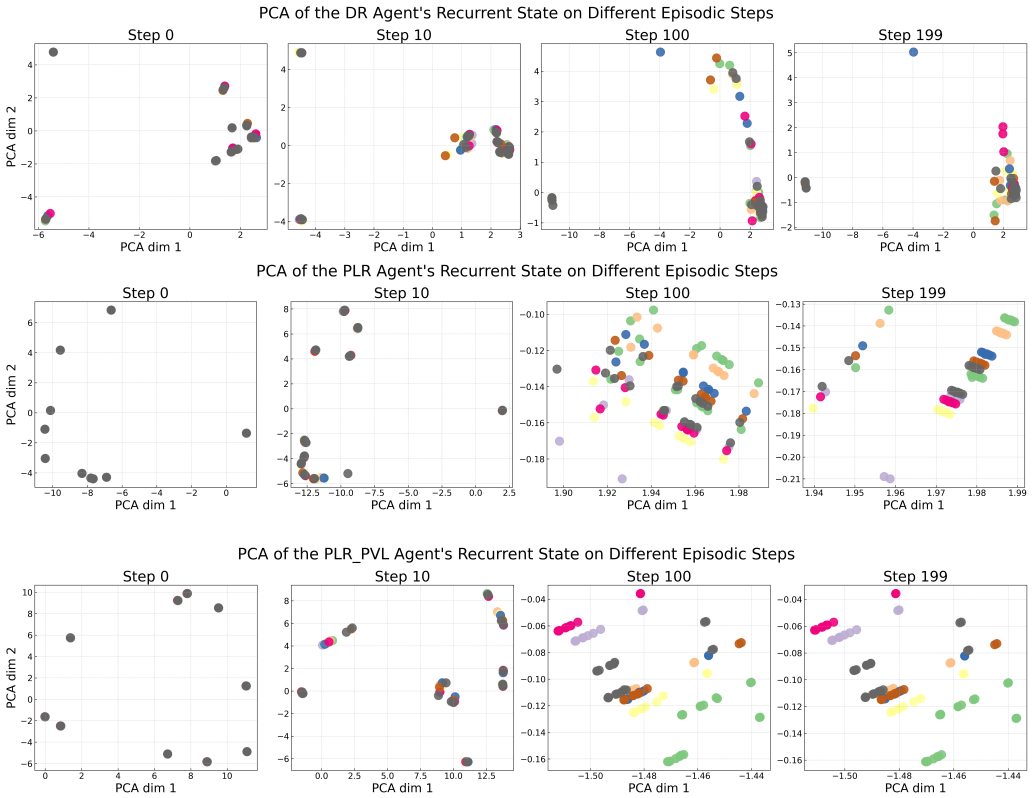


Figure 12: PCA of the hidden recurrent state for agents trained on different algorithms

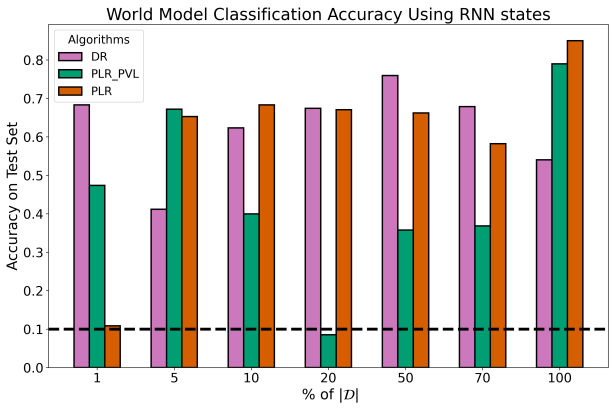
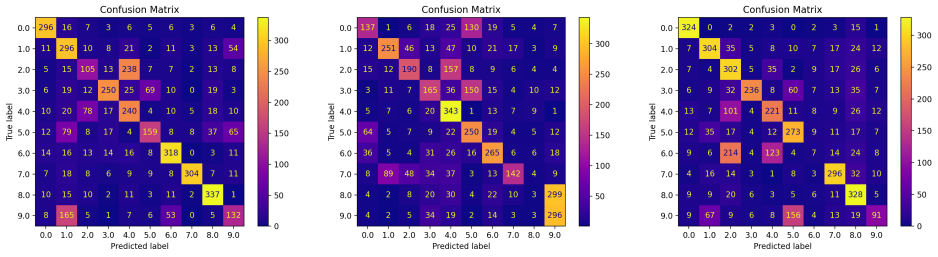


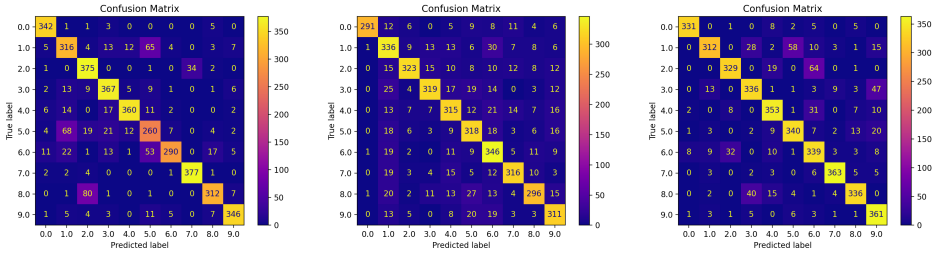
Figure 13: Classification accuracy of the hidden states from agents trained with DR, PLR, and PLR_PVL for a dataset of trajectories from 9 world models and the real environment. The dashed black line is the random prediction accuracy for the 10 classes.



(a) DR on 5% of $|\mathcal{D}|$

(b) DR on 50% of $|\mathcal{D}|$

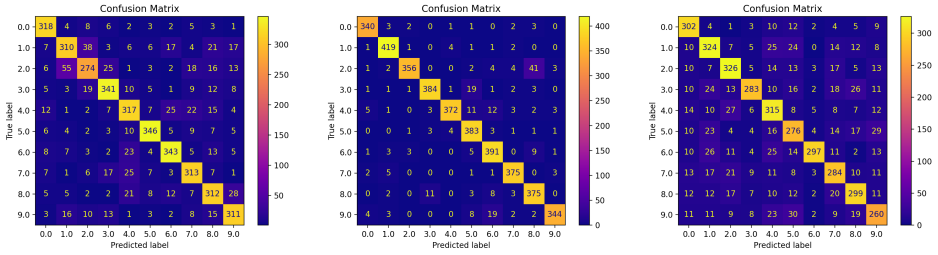
(c) DR on 100% of $|\mathcal{D}|$



(d) PLR on 5% of $|\mathcal{D}|$

(e) PLR on 50% of $|\mathcal{D}|$

(f) PLR on 100% of $|\mathcal{D}|$



(g) PLR_PVL on 5% of $|\mathcal{D}|$

(h) PLR_PVL on 50% of $|\mathcal{D}|$

(i) PLR_PVL on 100% of $|\mathcal{D}|$

Figure 14: Confusion Matrix for classifying 10 different levels or training environments using the RNN hidden states. Label 0 corresponds to the real **Pendulum** environment. Every row is a different training method where, DR is Domain Randomization, PLR is Prioritized Level Replay with an L_1 value loss score function and PLR_PVL refers to Prioritized Level Replay with an Positive Value Loss score function.

PART II

Transfer Report

Contents

List of Figures

1 Introduction

This report documents my progress as a probationary research student and demonstrates my readiness to transfer to DPhil status. My research aims to make Reinforcement Learning more applicable to real-world scenarios. The initial inspiration came from soft robotics challenges, but my work has evolved to address fundamental Machine Learning questions about learning from limited real-world data and ensuring reliable transfer of learned policies to physical systems.

My first research contribution was accepted at the Robosoft 2024 conference [Berdica et al., 2024a] and presented a novel approach using a limited dataset collected from a pneumatically actuated soft manipulator to develop a recurrent dynamics model. This model served as a Reinforcement Learning environment for training a goal-conditioned policy. While the dynamics model proved unstable and required separate policies for different regions of the manipulator's Cartesian space, with rewards computed as a function of goal distance, the work received valuable feedback. Specifically, reviewers at the Robosoft conference highlighted the innovative use of fully offline rollouts with state-of-the-art online learning algorithms.

Although limited in scope, this initial work served as an important learning experience in offline Reinforcement Learning and environmental modeling, building on the foundational World Models framework introduced by Ha and Schmidhuber [2018]. This experience naturally led to deeper investigation of offline rollouts

in World Models and methods for ensuring successful policy transfer to real environments. The results of this investigation appeared in my NeurIPS workshop paper on *Robust Offline Learning using World Models* Berdica et al. [2024b], with an expanded version currently under review at The International Conference of Learning Representations, as detailed on page ix in **Part I** of this report.

Beyond my first-author work, I contributed as a middle author to *DARE: The Deep Adaptive Regulator for Control of Uncertain Continuous-Time Systems* Waldon et al. [2024], which was accepted at the ICML Workshop on Foundations of Reinforcement Learning and Control. As the paper received a borderline rejection from NeurIPS, we are currently implementing reviewer suggestions and testing more challenging environments before an arxiv release and planned resubmission to ICML 2025.

To support the broader research community, I am preparing to release a completed JAX implementation of a one-step predictive dynamic model along with standalone data collection scripts like I did with `https://github.com/uljad/DaJax` [Berdica, 2024].

2
Literature Review

Contents

Here, I outline the relevant literature underpinning the soft robotics work and the Reinforcement Learning (RL) contributions. These sections are extension of the literature review conducted for the papers in **Part I**.

2.1 Reinforcement Learning

2.1.1 Markov Decision Process

A finite horizon Markov Decision Process (MDP) forms the mathematical foundation for reinforcement learning. The MDP is defined by the tuple $\langle \mathcal{S}_0, \mathcal{S}, \mathcal{A}, T, \mathcal{R}, I, \mathcal{O} \rangle$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, and $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ defines the transition dynamics as a distribution over next states given an initial

state and action.$\Delta(\mathcal{X})$ is the set of all probability distributions over the set $\mathcal{X}$. The initial state distribution is denoted by $\mathcal{S}_0$. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ outputs scalar rewards, where $r_t = \mathcal{R}(s_t, a_t)$ represents the reward at time t for action a_t taken in state s_t.

The observation function $\mathcal{I} : \mathcal{S} \to \Delta(O)$ maps states to observations. In fully observable environments, states and observations are equivalent. At time t, an agent observes o_t and takes action a_t, building an action-observation history $\tau_t = \langle o_0, a_0, ..., o_{t-1}, a_{t-1}, o_t \rangle$. A policy π maps observations in $\mathcal{O}$ to action distributions in $\mathcal{A}$. For recurrent policies, actions are conditioned on the entire history: $a_t \sim \pi(\tau_t)$. The policy is trained to maximize the discounted episodic return J_M^π for MDP M with fixed episode length T:

$$J_M^\pi := \mathbb{E}_{a_{0:T} \sim \pi, s_0 \sim \mathcal{S}_0, s_{1:T} \sim T} \left[\sum_{t=0}^{T} r_t \right]. \tag{2.1}$$

2.1.2 Offline Reinforcement Learning

Offline reinforcement learning methods aim to maximize J^π using a dataset $\mathcal{D}$ containing transition tuples $\{(s_n, a_n, r_n, s_{n+1})\}_{n=1}^N$ with initial states $s_0 \sim \mathcal{S}_0$. This dataset is collected by deploying a behavior policy π_b in the environment. While π_b may have varying levels of expertise and exploration, resulting in potentially incomplete state coverage, offline RL algorithms are designed to produce policies that achieve meaningful results in the true environment.

2.1.3 World Models

World models, as defined by Ha and Schmidhuber [2018], provide compact representations of environment dynamics independent of agent interactions. A one-step predictive world model assumes Markovian dynamics, represented by a neural network $\mathcal{F}_\theta$ with parameters θ. The model predicts transitions as $\hat{o}_{t+1}, \hat{r}_{t+1} \leftarrow \mathcal{F}_\theta(\hat{o}_t, a_t)$, generating both observations and rewards.

Agents trained exclusively in world models may exploit out-of-distribution states or discontinuities absent in real environments [Levine et al., 2020, Ha

and Schmidhuber, 2018]. Like other model-based RL methods, these models can accumulate compounding errors where predictions increasingly diverge from reality Saleh et al. [2022]. However, with proper implementation, agents can achieve strong performance even when training on full-length episodes entirely within world models.

2.1.4 Unsupervised Environment Design

Unsupervised environment design (UED) represents a class of autocurriculum methods where an adversary proposes tasks (or levels) for agent training. A common approach [Dennis et al., 2020] models environments as an Underspecified Partially Observable Markov Decision Process (UPOMDP) $\langle \mathcal{S}_0, \mathcal{S}, \mathcal{A}, T, \mathcal{R}, I, \mathcal{O}, \Theta \rangle$.

In this framework, Θ represents the set of free parameters that can adjust environment characteristics, with θ denoting a specific level. For example, Θ might define possible block placements in a grid [Chevalier-Boisvert et al., 2023], while θ specifies exact locations. Each θ creates a concrete POMDP.

Recent work has shown Minimax Regret (MMR) UED as an effective approach for training robust agents [Dennis et al., 2020, Jiang et al., 2021b,a, Parker-Holder et al., 2022]. The adversary selects levels that maximize agent regret, defined as $U_\theta(\pi^*_\theta) - U_\theta(\pi)$, where U represents discounted returns for a policy on level θ, and π^*_θ is the optimal policy for that level.

Dennis et al. [2020] formulated UED as a two-player, zero-sum game between adversary and policy. When the adversary maximizes regret and reaches Nash equilibrium with the policy, the policy satisfies:

$$\pi \in \arg\min_{\pi \in \Pi} \{\arg\max_{\theta \in \Theta} \{\text{Regret}_\theta(\pi)\}\}. \tag{2.2}$$

This equation shows that the policy minimizes its worst-case regret, providing robustness by bounding regret across all levels $\theta \in \Theta$.

2.1.5 Prioritized Level Replay

Prioritized Level Replay (PLR) [Jiang et al., 2021b] offers an empirically successful curriculum method based on curating high-scoring levels. PLR maintains a buffer of previous high-scoring levels and alternates between sampling from this buffer and sampling new levels. After rolling out the agent on these levels, they are scored based on agent performance. High-scoring levels are added to the buffer for future training. While sampling new levels is common, some applications may use a predefined, fixed level set [Tzannetos et al., 2024].

The original PLR implementation scores each level θ_i using a time-averaged L_1 value loss from the agent's last trajectory. To achieve minimax robustness, scoring functions should consider regret as described in the UED section. Jiang et al. [2021a] propose various scoring functions that better approximate regret, though the final choice depends on environment characteristics.

2.2 Soft Robotics

Soft robotic manipulators are made of compliant (i.e. flexible) material and exhibit a low Young's modulus that enables them to be arranged in highly deformable geometries Laschi et al. [2016]. These designs, inspired by boneless biological organisms, can undergo large elastic deformation throughout operations and facilitate safer interaction with the environments compared to their traditional rigid counterparts George Thuruthel et al. [2018]. The morphological dexterity outsources parts of the solution computation to the compliant material Hauser et al. [2011], but remains underactuated as the states of the physical body are governed by highly nonlinear continuum dynamics resulting in infinite degrees of freedom (DOF) being controlled by a finite number of actuators Laschi et al. [2023]. Given the inherent challenges, making use of the unique properties of soft robotics remains an open problem.

The existing analytical methods for accurate dynamic models in classical optimal control make reductive assumptions like constant-curvature and valve control

heuristics for trajectory optimization George Thuruthel et al. [2018] while relying on the material properties being unchanged or otherwise predictably modeled. These methods strive to balance the trade-off between reducing the complexity of the continuum dynamics and suppressing the undermodeled adaptive behavior that emerges through the soft robot's interaction with the environment.

Deep learning-based approaches do away with such reductions by utilizing the sensory data from the robot's interaction with the environment and directly modeling a nonlinear system. This allows for the learned models to make use of the morphological changes occurring in operation time. However, the use of learned (black-box) mappings between actuation and task space comes at an increased computational cost both during training and at test time. Learned control policies approaches that seek to map desired task space results to required inputs are either open loop due to the computational overhead Satheeshbabu et al. [2019], Thuruthel et al. [2017] or employ trajectory optimization and supervised learning with Guided Policy Search Levine and Koltun [2013] to derive the control policies from learned recurrent representations of the dynamic space Thuruthel et al. [2019]. The most recent work to date using closed loop policy gradients leverages Cosserat Rod Models simulations and short recurrence horizon for the learned forward dynamics Alessi et al. [2023]. Furthermore, these methods hinge on the limited number of recorded interactions in the dataset and existing prior knowledge of material heuristics Satheeshbabu et al. [2019] or of how to reach defined goals in the task space Rolf et al. [2010].

2.2.1 Modeling Challenges

Soft robotics controllers need to account for how robots adapt when interacting with complex environments, along with any behaviors that emerge from these interactions Hauser et al. [2011]. Traditional methods using parametric curve reconstructions don't consider the computational aspects inherent in the robot's physical form. This limitation makes data-driven models essential for understanding important insights from the resulting deformations Laschi et al. [2023].

For dynamic mapping, Thuruthel et al. [2019] uses a closed-loop guided policy search with a recurrent neural network (RNN) on discretized state transitions. Such mappings typically use non-linear autoregressive networks with exogenous inputs (NARX), with time delays ranging from one Thuruthel et al. [2019] to four Thuruthel et al. [2017], Alessi et al. [2023], Piqué et al. [2022]. We instead use a long short-term memory (LSTM) Hochreiter and Schmidhuber [1997] architecture through Heek et al. [2023] for faster performance. While both NARX and LSTM handle long-term dependencies, NARX networks use delayed connections that increase computational cost DiPietro et al. [2017] and make assumptions about actuation history's influence.

2.2.2 Reinforcement Learning Controllers

Reinforcement Learning (RL) has emerged as a powerful approach for sequential decision-making under uncertainty, particularly valuable when dealing with complex dynamic systems. Through repeated interactions with an environment, RL algorithms can learn effective control policies without requiring complete prior knowledge of system dynamics.

Early applications of RL to control problems relied heavily on value-based methods. For instance, Satheeshbabu et al. [2019] combined a Cosserat rod model with Deep Q-Learning for open-loop control policies. Similar Q-Learning approaches operating on discretized state spaces were explored by several researchers You et al. [2017], Satheeshbabu et al. [2019], Ansari et al. [2017]. The field then progressed toward actor-critic methods Ansari et al. [2017], with later work exploring different objectives such as minimizing control effort Thuruthel et al. [2019] and using PPO controllers with dynamic model simulations Alessi et al. [2023].

A significant advancement came with Proximal Policy Optimization (PPO) Schulman et al. [2017], which has proven particularly effective for continuous control problems Lillicrap et al. [2015], Van Hasselt [2012]. However, the increasing complexity of deep learning-based RL methods created new computational challenges. This led to the development of specialized frameworks focused on balancing hardware performance with user accessibility Hessel et al. [2021]. A notable recent innovation

is PureJaxRL Lu et al. [2022], which leverages JAX Bradbury et al. [2018] to execute both agents and environments on GPUs, achieving significant performance improvements over previous approaches.

3 Proposal

Contents

3.1 Bigger Picture Planning

The time to ideate and realize ground-breaking research is quite limited. At the start of the PhD, there is more time to learn and explore building workflows with close to no additional commitments. This setup changes as secondary authorship side projects become available due to the concentration of time and expertise overlapping with what the project in progress needs. The exploration and learning

time is further reduced by rebuttals preparations, the writing of this report and reviewing paper for different venues.

3.1.1 Reflecting on the past year

I believe that this medium is a suitable for a short reflection on areas of improvement. Providing a short analysis on the suboptimal aspect of my research conduct so far will help guide more efficient choices for the remainder of my time.

Working alone helps if the scope is well-defined. An insufficiently scoped work can expand over multiple conference cycles and isolate me from participating in larger and more ambitious collaborations or even starting new projects on my own.

The lack of scope should be compensated by an airtight and conservative addition of collaborators. An external researcher's interest in the project is as good as the interest in defining and being accountable for the assigned level of engagement ranging from method design, code writing and down to experimentation tracking and paper writing.

Clear and clean code is everything. The multi-year impact of a lot of machine learning contributions relies on the clarity and availability of the implementation. That matters as much as the paper getting accepted. I appreciate that this is not the case in many communities but it certainly is a good signal with the emergence of scale-enabling search methods as an acceptable contribution.

3.1.2 Project Selection

In my first year, I wanted to become a more well-rounded researcher and software engineer and willingly leaned into implementing my own pipelines and workflows which lead to exploring new methods under new constraints. I am also aware that a lot of publication success relies on coding in pairs and exploiting existing code bases. Reflecting on the limitation on my work last year, I am outlining the criteria in how I will choose my projects.

Expand my capacity as a developer A project should help me understand a new field better and create tools that will help me validate and shorten the

development time on immediate follow-up work, whether led by me or students in the same academic supervision circle.

Understand a potentially relevant field better Finding connections new in fields that are sparsely connected at best can be a very interesting source of novelty and pioneering work. For examples, growing my expertise in an LLM project can not only lead to a good scientific contribution published at a fitting venue, but also provide me with the tools and understanding to address questions closer to other field of interest in Reinforcement Learning.

Exploit existing tools A measured period of exploration and tinkering with existing tools and experimental pipelines may not always leads to pioneering and paradigm-shifting work but it is certainly a risk-averse way to contribute to the field and do an outer loop evaluation on the state-of-the-art and how it came to be.

3.2 Unified Offline RL Implementations

Offline RL is a very dispersed field of RL with a lot of benchmarks being either deprecated, saturated or both. The principle behind offline RL is that of training a policy from past trajectories collected by a behavior policy without any signal from the real environment in training time.

The field has recently stagnated with opaque hyperparameter sweeping protocols, implementations that are very different from each-other and make comparisons difficult and – with the exception of RAMBO by [Rigter et al., 2022] – pivotal implementation tricks that are not fully mentioned in the original paper. I aim to implement the most important model-free and model-based algorithms in a shared single-file format using JAX. Notable considerations include:

3.2.1 Termination Functions

Unified implementations like the one by Sun [2023] use hard-coded termination functions for each environment which assumes knowledge of the environment beyond the dataset. While the research community seems open to this trick, a more principled approach of either not terminating during unhealthy transitions Berdica

et al. [2024b] or training a binary classifier with an artificially balanced dataset would be a more principled approach that opens offline RL to broader use.

3.2.2 Hyperparameter Search

Hyperparameter sweep is literally swept under the rug and there are hyperparameters for every single environment and dataset. In classic model-based methods like MOPO, most implementations delegate entire algorithmic choices to hyperparameters like the reward standard deviation penalty between ensemble members, number of ensemble members and different methods of performing inference via the ensemble e.g. greedy, elites or random choice.

3.2.3 Related Work

There are three main efforts that aim to tackle the similar challenge.

OfflineRL-Kit by Sun [2023] implements a wide array of model-based and model-free algorithms but the implementation is on PyTorch, every evironment has carefully tuned hyperparameters and the implementation for each algorithm is spread across several files. This library does not feature more recent methods diffusion methods generating on-policy [Rigter et al., 2023] and off-policy [Jackson et al., 2024] trajectories.

CORL [Tarasov et al., 2022] containes single file implementation of *only* model-free algorithms in the style of CleanRL [Huang et al., 2022]. **JAX-CORL** [Nishimori, 2024] is a line by line conversion of CORL's PyTorch operations to JAX which does underutilizes JAX functionalities.

3.2.4 Current Progress

At the time of writing, the finished implementations with scores matching and exceeding the reported values include the entirety of model-free methods SAC with a diverse ensemble of Q-networks [An et al., 2021], Conservative Q-Learning

(CQL) [Kumar et al., 2020], Behavior Cloning (BC), Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [Fujimoto et al., 2018] and TD3+BC [Fujimoto and Gu, 2021].

Model-based additions that are fully implemented but are yet to match the reported results from other implementations are Model-Based Offline Policy Optimization (MOPO) [Yu et al., 2020] and the halting function addition used in MOREL Kidambi et al. [2020]. My collaborator will also add the very recent yet well-received Policy-Guided Diffusions (PGD) algorithm Jackson et al. [2024].

My fully method using fully offline rollouts on World Models trained by shuffled transitions lends itself easily to this implementation paradigm as everything in implemented in JAX, including the data collection code from any environment outside common benchmark datasets like D4RL [Fu et al., 2020] and its potential successor, Minari [Younis et al., 2024].

Once we verify these implementations and ensure every useful addition is transparently marked by simply comparing the files line by line, the unified library of Unified Offline RL Implementations (**Unifloral**) will be made public for the wider community. I plan to add RAMBO [Rigter et al., 2022] after the ICML deadline as it is the highest performing and more robust one to date, including the original implementation.

Additional publication plans include a position paper for ICML and then proposing a fair evaluation methodology between the methods that takes into account all the resources used to train the model and tune the hyperparameters. The latter would most likely be a better fit for the Reinforcement Learning Conference (RLC) or NeurIPS.

3.3 Intent Factored generation using LLMs

Intent Factored Generation is a novel approach to controlling language model outputs by separating semantic content from stylistic variation. The method first extracts semantic control tokens from desired responses using a pretrained LLM, then finetunes another LLM to generate these tokens before producing the final

text. By using different sampling temperatures for the control tokens versus the response generation, this two-stage process enables the creation of responses that maintain semantic diversity while adhering to the original prompt's intent.

3.3.1 Connection to RL

I want to use the knowledge and the expertise built by working on this LLM project to conduct experiments on the unified implementation outlined in 3.2. I am very interested in making different parts of the algorithms and see how trivial each algorithmic improvement is or if other symbolic improvement can be made.

I am very interested in disambiguating scientific insights, algorithmic novelty and engineering prowess. The opaque interplay between these factors can create confusion regarding the long-term contributions of each work and guide the field in states of catastrophic forgetting or stagnant learning.

3.4 Useful Representations in RL

Ingebrand et al. [2024] introduced a zero-shot reinforcement learning approach that encodes task reward functions into a latent space used by the policy. Their functional encoder represents reward functions as linear combinations of learned neural network basis functions. Specifically, given a set of functions $\mathcal{F}$, the encoder learns basis functions $g^{\theta_1}, ..., g^{\theta_K}$ by minimizing:

$$\theta^* \in \arg\min_{\theta} \sum_{f \in \mathcal{F}} \int \left| f(x) - \sum_{k=1}^{K} c_k(f) g_k^{\theta}(x) \right|^2 dx \tag{3.1}$$

$$\text{s.t.} c_k(f) = \int f(x) g_k^{\theta}(x) dx \tag{3.2}$$

Let $\mathbf{c}_f = (c_1(f), ..., c_K(f))$ where $c_{1:K}$ are the coefficients for the basis functions. In Ingebrand et al. [2024], $\mathcal{F}$ corresponds to a set of transition (or reward) functions, and the agent is given a dataset of samples of each function. After training the encoder, an RL agent is trained on environments $f \in \mathcal{F}$, learns a policy $\pi := \pi(s; \mathbf{c}_f)$.

One drawback of this method, which is acknowledged by the authors, is that "a small change in the [reward] function can sometimes lead to abrupt and discontinuous

changes in the optimal policy." Hence, a good latent representation $E(f)$ should have the property that if for $f_1, f_2 \in \mathcal{F}$ with π_{f_1}, π_{f_2} the optimal policies:

$$\pi_{f_1} \approx \pi_{f_2} \implies E(f_1) \approx E(f_2)\,. \tag{3.3}$$

Motivated by the description of scalable zero-short RL by Touati and Ollivier [2021], I want to implement a *policy-informed functional encoding* which learns a latent representation reward functions that depends on the policy which optimizes the given reward function. This will be achieved through an auto-encoder f^θ, which is trained through following loss:

$$\mathcal{L}(f;\theta) = R(f;\theta) + C(f;\theta)\,, \tag{3.4}$$

where $R(f;\theta)$ denotes the reconstruction loss of a function f and $C(\pi_f;\theta)$ is a contrastive loss defined in the following way. Let $E(f)$ denote the encoding of f. For each π_f such that $f \in \mathcal{F}$, say that $f' \in \mathcal{F}$ is a *positive example* if $d(\pi_f, \pi_{f'}) < \epsilon$ for some distance metric (e.g. KL) and a *negative example* otherwise. Then we take

$$C(f;\theta) = \max\left(0, |E(f) - E(f^+)|^2 - |E(f) - E(f^-)|^2 + \epsilon\right)\,, \tag{3.5}$$

for some positive and negative examples f^+ and f^-.

This addresses the problem outlined in Ingebrand et al. [2024] where a small change in the reward function can lead to very different policies. Similar work by Frans et al. [2024] introduces a Functional Reward Encodings approach where they use a transformer to learn latent representations of real trajectories and random reward functions. These representations are used to train a general policy which can perform well in real environment with the non-random reward function not included in the supervised training of the encoder.

Given the existing clear implementation and my familiarity with the supervised learning methods used in this body of work, I expect to make a submission to an ICLR workshop and then to NeurIPS. Given the reviewers' requirement for the rebuttals, it is withing the realm of possibility to finalize the first batch of experiments and consider an ICML submission in February.

3.5 Deploying on Soft Robots

The challenge stands in writing code to load the JAX weights on a ROS-compatible interface that will allow the controllers to run in real-time and even train in a safe way using the methods in the Berdica et al. [2024a,b].

3.6 Planned Timeline

My timeline is summarized in Figure 3.1 as a Gantt chart. This is used to visualize how different projects and papers will overlap and a tentative timeline to ensure well-documented contributions, complete code releases and accepted papers in peer-reviewed, high-impact venues.

The time from the submission of this report until the end of December 2024 will be dedicated to finalizing the camera-ready version of the ICLR submission for *arxiv*, independent of the final acceptance decision. I will also continue the ICML projects outlined in the charts below and open-source the tools I have developed in my first year with proper announcements and social network campaigns.

Due to the page dimensions and the cyclical nature of the robotics and machine learning review cycles, I have separated the chart in multiple parts for better legibility.

The NeurIPS and the following year is less detailed due to their dependence on the results leading up to the NeurIPS mobilization.

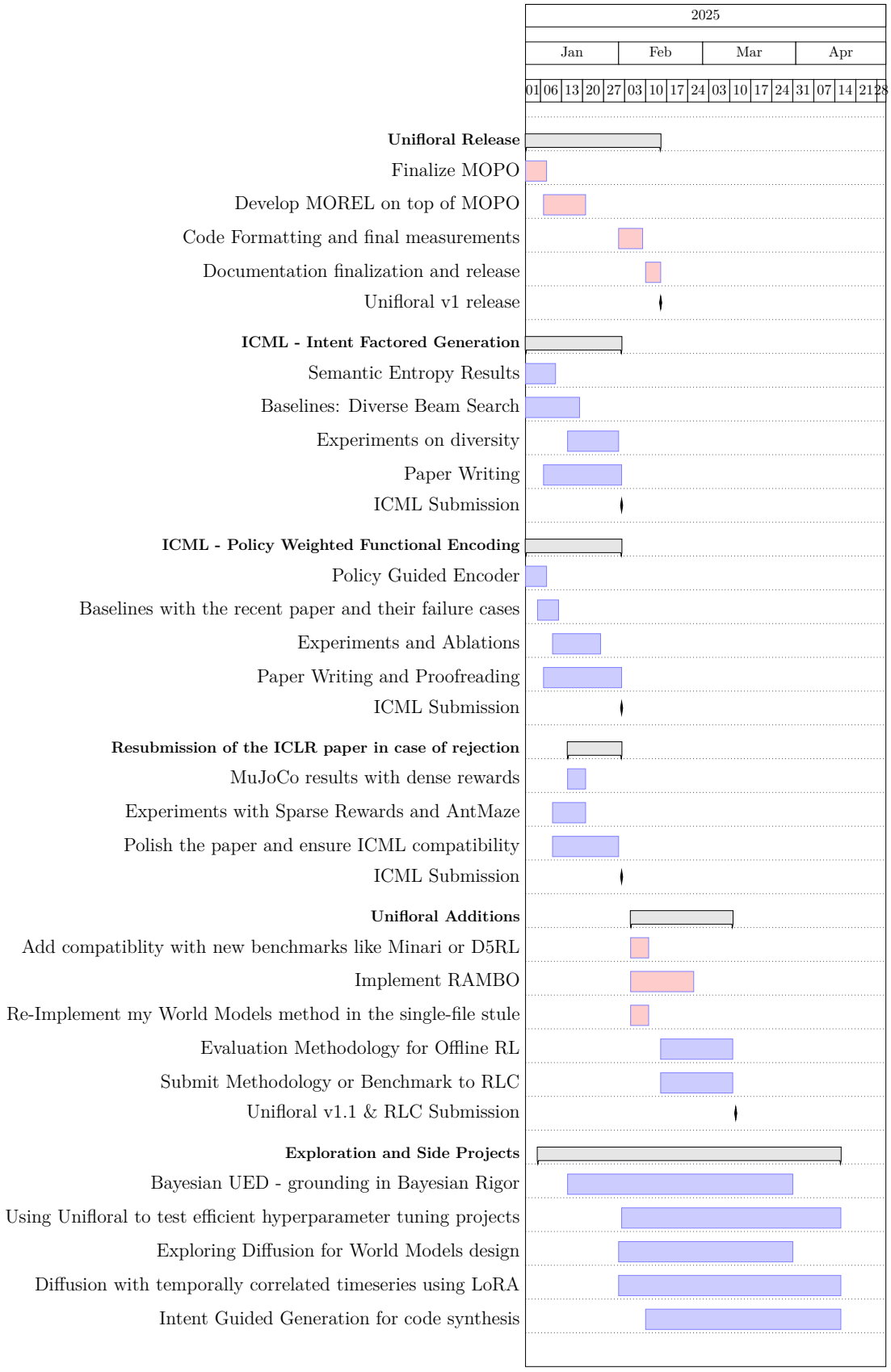


Figure 3.1: Gantt Chart leading up to the NeurIPS focus

Figure 3.2: Gantt Chart for the Second Half of 2025

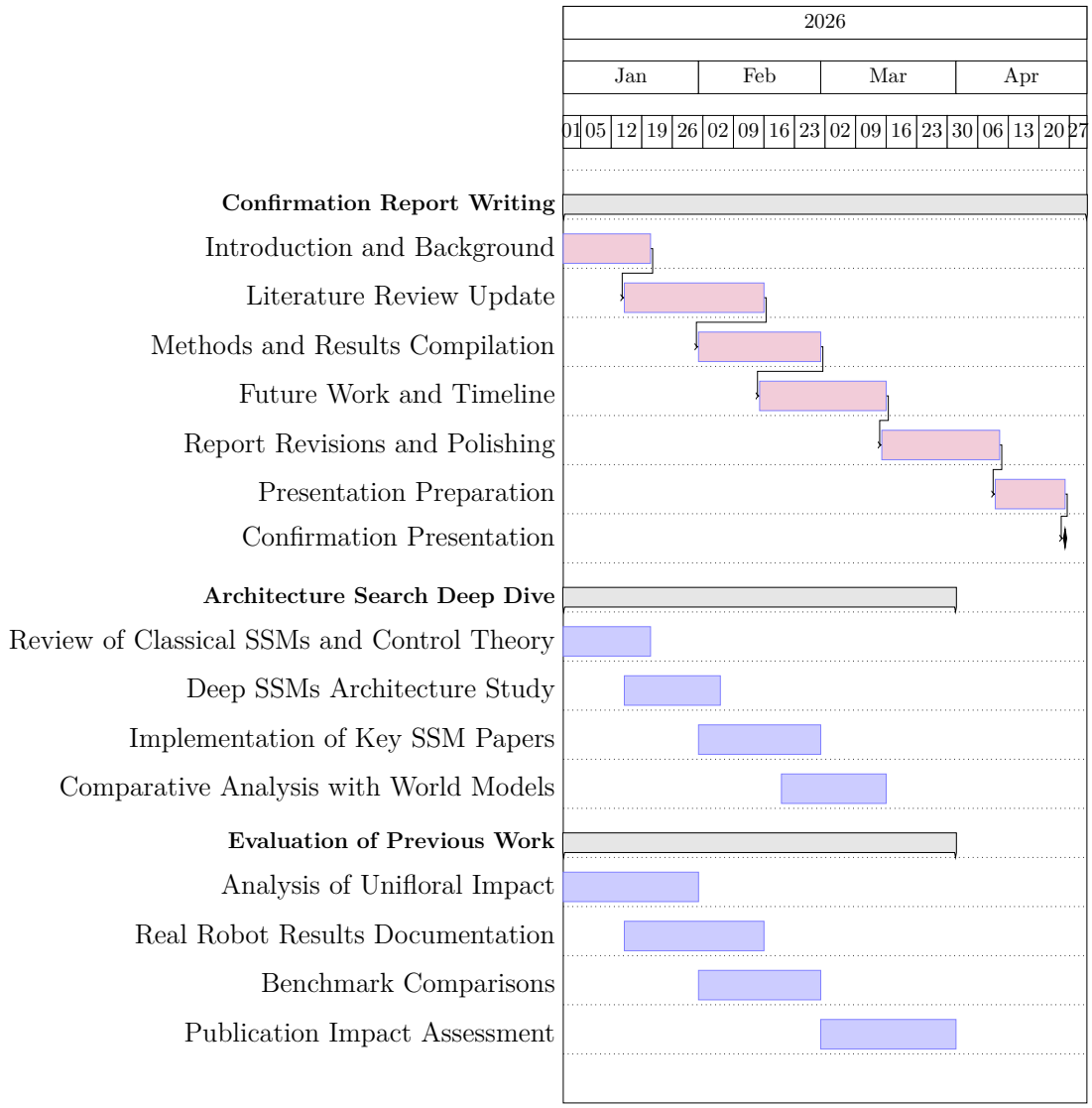


Figure 3.3: Gantt Chart for Confirmation Preparation Period

4 Acknowledgments

Personal

I am deeply grateful to my family for their unwavering support and endless patience throughout this academic journey.

I would like to express my sincere gratitude to my lab mates who have become both colleagues and friends. The countless hours of collaboration, debugging sessions, and shared experiences have made this work not just possible, but enjoyable.

Institutional

The inclusive environment at Oxford has been fundamental to my growth as a researcher. The diverse perspectives, open dialogue, and supportive community have enriched both my research and personal development.

A special thank you goes to Wendy Poole, whose exceptional work as the CDT AIMS administrator has been instrumental to my research journey. Her tireless efforts in managing travel logistics, administrative complexities, and day-to-day operations have allowed me and my colleagues to focus on our scientific pursuits. Wendy's proactive approach, understanding nature, and remarkable efficiency in handling any challenge that comes her way have created an environment where researchers can truly thrive.

Bibliography

Carlo Alessi, Helmut Hauser, Alessandro Lucantonio, and Egidio Falotico. Learning a controller for soft robotic arms and testing its generalization to new observations, dynamics, and tasks. In *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, pages 1–7. IEEE, 2023.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Yasmin Ansari, Mariangela Manti, Egidio Falotico, Yoan Mollard, Matteo Cianchetti, and Cecilia Laschi. Towards the development of a soft manipulator as an assistive robot for personal care of elderly people. *International Journal of Advanced Robotic Systems*, 14(2):1729881416687132, 2017.

Uljad Berdica. Dajax: Data collection in the jax ecosystem, 2024. URL `https://github.com/rodrigodelazcano/DaJax`.

Uljad Berdica, Matthew Jackson, Niccolò Enrico Veronese, Jakob Foerster, and Perla Maiolino. Reinforcement learning controllers for soft robots using learned environments. In *2024 IEEE 7th International Conference on Soft Robotics (RoboSoft)*, pages 933–939. IEEE, 2024a.

Uljad Berdica, Kelvin Li, Michael Beukman, Alexander David Goldie, Perla Maiolino, and Jakob Nicolaus Foerster. Robust offline learning via adversarial world models. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024b.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

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. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Robert DiPietro, Christian Rupprecht, Nassir Navab, and Gregory D Hager. Analyzing and exploiting narx recurrent neural networks for long-term dependencies. *arXiv preprint arXiv:1702.07805*, 2017.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. *arXiv preprint arXiv:2402.17135*, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34: 20132–20145, 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, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/fujimoto18a.html`.

Thomas George Thuruthel, Yasmin Ansari, Egidio Falotico, and Cecilia Laschi. Control strategies for soft robotic manipulators: A survey. *Soft robotics*, 5(2): 149–163, 2018.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Helmut Hauser, Auke J Ijspeert, Rudolf M Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological cybernetics*, 105:355–370, 2011.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

Matteo Hessel, Manuel Kroiss, Aidan Clark, Iurii Kemaev, John Quan, Thomas Keck, Fabio Viola, and Hado van Hasselt. Podracer architectures for scalable reinforcement learning. *arXiv preprint arXiv:2104.06272*, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

Tyler Ingebrand, Amy Zhang, and Ufuk Topcu. Zero-shot reinforcement learning via function encoders. *arXiv preprint arXiv:2401.17173*, 2024.

Matthew Thomas Jackson, Michael Tryfan Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and Jakob Foerster. Policy-guided diffusion. *arXiv preprint arXiv:2404.06356*, 2024.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021b.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020.

Cecilia Laschi, Barbara Mazzolai, and Matteo Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science robotics*, 1(1): eaah3690, 2016.

Cecilia Laschi, Thomas George Thuruthel, Fumiya Lida, Rochdi Merzouki, and Egidio Falotico. Learning-Based Control Strategies for Soft Robots: Theory, Achievements, and Future Challenges. *IEEE Control Systems Magazine*, 43 (3):100–113, June 2023. ISSN 1941-000X. doi: 10.1109/MCS.2023.3253421. Conference Name: IEEE Control Systems Magazine.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1–9. JMLR.org, 2013. URL `http://proceedings.mlr.press/v28/levine13.html`.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 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. *arXiv preprint arXiv:1509.02971*, 2015.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Soichiro Nishimori. Jax-corl: Clean sigle-file implementations of offline rl algorithms in jax. 2024. URL `https://github.com/nissymori/JAX-CORL`.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022. URL `https://proceedings.mlr.press/v162/parker-holder22a.html`.

Francesco Piqué, Hari Teja Kalidindi, Lorenzo Fruzzetti, Cecilia Laschi, Arianna Menciassi, and Egidio Falotico. Controlling soft robotic arms using continual learning. *IEEE Robotics and Automation Letters*, 7(2):5469–5476, 2022.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

Marc Rigter, Jun Yamada, and Ingmar Posner. World models via policy-guided trajectory diffusion. *arXiv preprint arXiv:2312.08533*, 2023.

Matthias Rolf, Jochen J Steil, and Michael Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

Esra' Saleh, John D Martin, Anna Koop, Arash Pourzarabi, and Michael Bowling. Should models be accurate? *arXiv preprint arXiv:2205.10736*, 2022.

Sreeshankar Satheeshbabu, Naveen Kumar Uppalapati, Girish Chowdhary, and Girish Krishnan. Open loop position control of soft continuum arm using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5133–5139. IEEE, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yihao Sun. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. `https://github.com/yihaosun1124/OfflineRL-Kit`, 2023.

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`.

Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. Learning dynamic models for open loop predictive control of soft robotic manipulators. *Bioinspiration & biomimetics*, 12(6):066003, 2017.

Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators. *IEEE Transactions on Robotics*, 35(1):124–134, February 2019. ISSN 1941-0468. doi: 10.1109/TRO.2018.2878318. Conference Name: IEEE Transactions on Robotics.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Georgios Tzannetos, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum with task correlations for deep reinforcement learning. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 5027–5036. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/556. URL `https://doi.org/10.24963/ijcai.2024/556`. Main Track.

Hado Van Hasselt. Reinforcement learning in continuous state and action spaces. In *Reinforcement Learning: State-of-the-Art*, pages 207–251. Springer, 2012.

Harrison Waldon, Fayçal Drissi, Yannick Limmer, Uljad Berdica, Jakob Nicolaus Foerster, and Alvaro Cartea. Dare: The deep adaptive regulator for control of uncertain continuous-time systems. In *ICML 2024 Workshop: Foundations of Reinforcement Learning and Control–Connections and Perspectives*, 2024.

Xuanke You, Yixiao Zhang, Xiaotong Chen, Xinghua Liu, Zhanchi Wang, Hao Jiang, and Xiaoping Chen. Model-free control for soft manipulators based on reinforcement learning. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2909–2915. IEEE, 2017.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL `https://doi.org/10.5281/zenodo.13767625`.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.