Learning In-Context Decision Making with Synthetic MDPs

Akarsh Kumar
MIT
akarshkumar0101@gmail.com

Chris Lu
Oxford
christopher.lu@eng.ox.ac.uk

Louis Kirsch
IDSIA / USI / SUPSI
mail@louiskirsch.com

Phillip Isola
MIT
phillipi@mit.edu

Abstract

Current AI models are trained on huge datasets of real world data. This is increasingly true in RL, with generalist agents being trained on data from hundreds of real environments. It is thought that real data/environments are the only way to capture the intricate complexities of real world RL tasks. In this paper, we challenge this notion by training generalist in-context decision making agents on only data generated by simple random processes. We investigate data generated from eight different families of synthetic environments ranging from Markov chains and bandits to discrete, continuous, and hybrid Markov decision processes (MDPs). Surprisingly, the resulting agents' performances are comparable to agents trained on real environment data. We additionally analyze what properties of the pretraining MDPs are ideal for creating good agents, thus giving RL practitioners insights on choosing which environments to train on.

1 Introduction

Recent AI advances, particularly in computer vision and natural language processing, are largely driven by large datasets [Kaplan et al., 2020]. Although the same scale models and data have not yet been achieved in reinforcement learning (RL), the field is anticipating a similar data-driven approach going forward [Baker et al., 2022, Reed et al., 2022, Team et al., 2023]. The goal is to create a general-purpose decision making agent by training it on thousands of real environments, since, intuitively, the only way to capture the complexities of real environments is to train on them directly. However, one major bottleneck in RL is the creation of thousands of environments and the data collection process within them. Creating environments requires either constructing real world setups or programming high quality simulators/games to encapsulate real world tasks. Collecting data within these environments is also extremely challenging because of the difficulty in scaling online RL and collecting human demonstrations. In this paper, we challenge the assumption that real environments/datasets are required for the problem of decision making by asking how far can one get with only data generated from simple processes.

We create eight different families of synthetic-MDPs: Markov chains, bandits, discrete MDPs, multi-discrete MDPs, continuous MDPs, hypersphere continuous MDPs, discrete-embedded continuous MDPs, and discrete-continuous hybrid MDPs. Each family contains different axes of variation such as the initial state distribution variance, transition complexity, transition stochasticity, etc. Expert data is then collected on each of these distributions of environments. Following Kirsch et al. [2022b] we augment this synthetic data with random projections to increase data diversity. This data is used

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

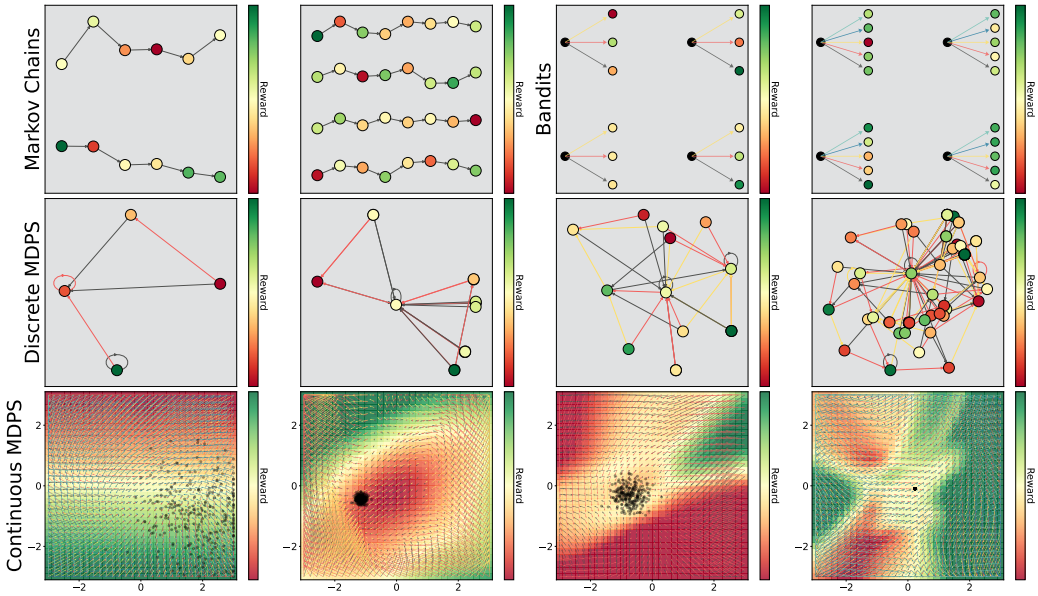


Figure 1: Visualizations of some of our synthetic MDPs. In the discrete environments, nodes represent possible states. In the continuous MDPs the entire 2-D space is the state space. States are colored by reward values. Arrows show the transition function and arrow colors represent the possible actions at any state. In the continuous MDPs, the black circles show the initial state distribution.

to train a general in-context decision making agent that efficiently imitates expert actions. The agent is then evaluated on its transfer performance on real RL environments of interest.

Surprisingly, agents trained on only synthetic data perform as well as agents trained on real environments. This finding questions the true complexity of the problem of in-context imitation learning - if simple processes produce high-performance decision makers, then imitation learning may be simpler than expected. Because of the simplicity of constructing synthetic MDPs and collecting data within them, this paradigm provides a natural direction for scaling RL agents.

We perform several ablations on the best synthetic MDP distribution to investigate what properties are important for transfer performance. Our analysis confirms expected results, such as the positive impact of nonlinear transition complexity, while also revealing unexpected insights, such as the advantage of initial state variance. We believe this analysis can help RL practitioners create and select training environments for better transfer performance.

2 Related Works

2.1 Synthetic Data/Environments

Though relatively new, there is a line of work on using purely synthetic data for training AI models [Oh et al., 2020, Kataoka et al., 2021, Müller et al., 2022]. In vision, Kataoka et al. [2021], Baradad et al. [2022, 2023], show that simple processes can create synthetic images which are good pretraining data for transfer to real images. In language, a domain thought to be immensely complex, Krishna et al. [2021], Ri and Tsuruoka [2022], Chiang and yi Lee [2022] shows similar observations. In reinforcement learning, besides the extensive use of simulators and games, which can be considered a form of synthetic data, there is much less exploration on very simple processes for training. For instance, Oh et al. [2020] generate distributions of grid world environments for gradient-based meta-learning and Miconi [2023] describe many different properties synthetic MDPs may exhibit. Ferreira et al. [2021, 2022], Liesen et al. [2024]) propose to *meta-learn* the synthetic task such that training on it results in agents capable of solving a target real task. This approach is akin to dataset distillation [Wang et al., 2020b] and uses real environment data rather than training purely from scratch. In other works existing datasets or environments are heavily randomized with synthetic projections in a way

that leads to in-context learning [Kirsch et al., 2022b, Lu et al., 2024, Kirsch et al., 2023]. In contrast, our work focuses on the generation and analysis of the synthetic environments themselves.

2.2 Generalist Agents/Transfer in RL

For most of the history of RL, the field has evaluated RL agents on the same singleton environments they were trained on [Kirk et al., 2023, Bellemare et al., 2013], due to the difficulty of transfer learning in RL. Only recently are RL agents being trained and evaluated on distributions of environments [Kirk et al., 2023, Cobbe et al., 2020, Küttler et al., 2020, Reed et al., 2022]. Even then, there are levels and axes of generalization, IID, OOD, new initial states, rewards, dynamics, etc [Kirk et al., 2023]. Most previous works change the simpler axes of generalization like initial states and reward, but rarely change dynamics or the entire environment domain [Kirsch et al., 2020, Cobbe et al., 2020, Küttler et al., 2020, Team et al., 2021, 2023, Laskin et al., 2022]. Most previous works which tackle hard OOD generalization to completely new domains fallback to doing gradient adaptation for evaluating transfer performance [Reed et al., 2022, Lee et al., 2022, Reid et al., 2022, Wang et al., 2024]. While initial in-context meta-RL works focused mainly on relatively similar train-test environment distributions [Hochreiter et al., 2001, Wang et al., 2016, Duan et al., 2016, Laskin et al., 2022] there is a growing literature that explores out-of-distribution generalization to new domains without the use of gradients [Kirsch et al., 2022a, 2023, Lu et al., 2024, Raparthy et al., 2023]. We continue the exploration of such out-of-distribution generalization by relying solely on synthetic environments instead of real environments.

3 Methods

3.1 Shared Observation/Action Space

Each environment $\mathcal{E}$ has an observation space $\mathbb{R}^{O_\mathcal{E}}$ and action space $\mathbb{R}^{A_\mathcal{E}}$. To create a generalist agent capable of transferring across environments, the agent must be able to accommodate observation and action spaces of varying sizes using a fixed-size neural network. Following (Kirsch et al. [2023], Lu et al. [2024]), we achieve this by projecting the observations and actions into a higher dimensional space using fixed environment-specific matrices. Environment observations, $o_\mathcal{E}$, are forward transformed with $o = P_\mathcal{E} o_\mathcal{E}$ where $P_\mathcal{E} \in \mathbb{R}^{O \times O_\mathcal{E}}$ is a fixed random semi-orthogonal matrix. Similarly, environment actions, $a_\mathcal{E}$, are forward transformed with $a = Q_\mathcal{E} a_\mathcal{E}$ where $Q_\mathcal{E} \in \mathbb{R}^{A \times A_\mathcal{E}}$ is a fixed random semi-orthogonal matrix. Importantly, the action projection can be inverted, such that actions can be transformed back to the native action-space at inference time. In our experiments, we choose the universal observation and action dimensions to be $O = 32$ and $A = 8$.

We convert all discrete action environments to continuous ones by re-defining the environment interface to receive continuous action vectors that are then interpreted as logits and softmaxed to generate the discrete probability distribution.

3.2 Synthetic MDPs

We construct eight different families of MDPs, mentioned below, ranging in different state and transition dynamics structures. A visualization of some of the MDP families is shown in Figure 1. For each family, we define up to 17 axes of variation in how the MDP can vary. Each point in this space of variations represents a *distribution* of environments. To generate expert data, we sample many environments and, for each environment, train specialist PPO Schulman et al. [2017] expert policies to collect data.

3.2.1 Synthetic MDP Families

Discrete MDP The discrete MDP family has a discrete state space and models the transition probabilities using a dense matrix. Discrete MDPs represent MDPs where the state space is small and finite and there is no underlying structure in the transitions, observations, etc. This family is denoted at "dsmdp" in all plots.

Discrete MDPs are implemented by keeping a dense $N \times A \times N$ matrix of probability logits describing transition dynamics, where A is the number of actions and N is the number of states. A softmax

temperature parameter controls the stochasticity of the transition dynamics. The observation mapping is also stored as a $N \times O$ matrix describing the observation at each state.

Markov Chain The Markov Chain family is identical to the discrete MDP family with the constraint that there is only one action and that reward is always zero. Markov Chains represent MDPs where trajectories are uncontrollable by the agent and solely determined by the environment. This family is denoted at "mchain" in all plots.

Bandit The bandit family is identical to the discrete MDP family with the constraint that the environment is reset to the initial state after every time step. This removes the effect of the transition dynamics. Bandits represent MDPs with no temporal aspect, resulting in greedy one-step problems. This family is denoted at "bandit" in all plots.

Multi-Discrete MDP The multi-discrete MDP family has a combinatorial discrete state space and models the transition probabilities using a randomly initialized neural network. Multi-discrete MDPs represent MDPs where the state space is large but discrete and there exists low dimensional structure in the transitions. This family is denoted at "mdsmdp" in all plots.

Multi-discrete MDPs are implemented with states which are M discrete numbers which can take on N possible values, leading to N^M possible states. The dynamics are constructed by concatenating the one-hot encoded states and actions and passing them to a random MLP to get an output which is then interpreted as M distributions over N values. A softmax temperature parameter controls the stochasticity of the transition dynamics. The transition complexity is controlled by the number of layers in the transition network.

Continuous MDP The continuous MDP family has a continuous state space and models the transition probabilities using using a randomly initialized neural network. Continuous MDPs represent MDPs with large smooth state spaces with underlying structure in the transitions. This family is denoted at "csmdp" in all plots.

Continuous MDPs are implemented with states which are D dimensional vectors. The dynamics are constructed by concatenating the state and the one-hot encoded action and passing them to a random MLP to get the next state. Gaussian noise is added to the state to control the stochasticity of the transition dynamics. The transition complexity is controlled by the number of layers in the transition network.

Continuous-Embedded MDP The continuous-embedded family is identical to the continuous MDP family with the constraint that the state vector must be snapped back on to the closest vector in a set of fixed embedding vectors. Continuous-embedded MDPs represent MDPs where the state space is small and finite and there is structure in the transitions. This family is denoted at "ecsmdp" in all plots.

Continuous-Hypersphere MDP The continuous-hypersphere family is identical to the continuous MDP family with the constraint that the state vector is normalized to unit norm every step. Continuous-hypersphere MDPs represent MDPs where the state space lies in a specific manifold within euclidean state space. This family is denoted at "ucsmdp" in all plots.

Hybrid MDP The hybrid family is composed of state space which is partly multi-discrete and continuous and models the transition probabilities using a randomly initialized neural network. Hybrid MDPs represent complex MDPs with heterogeneous dynamics structures. This family is denoted at "hsmdp" in all plots.

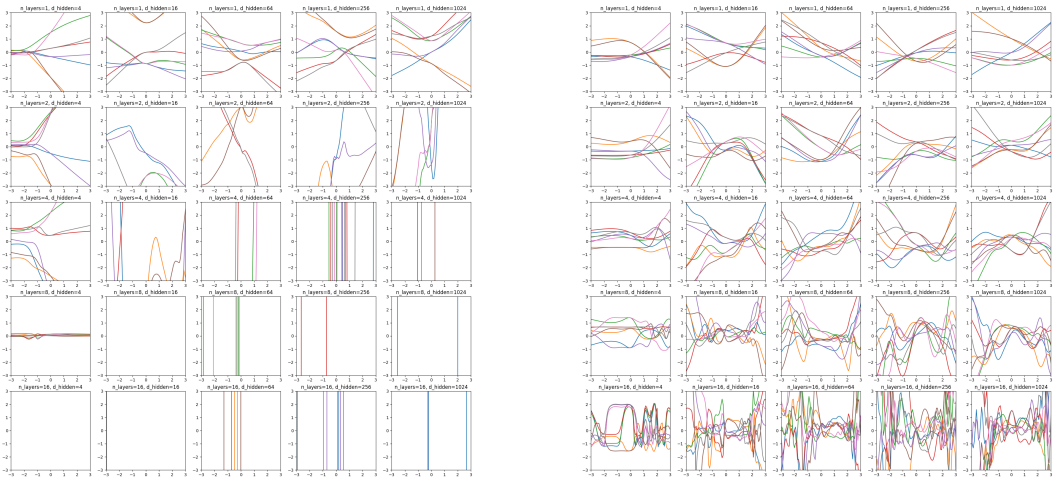
3.2.2 Axes of Variation in Synthetic MDPs

Because of the diverse nature of the families of synthetic MDPs, each axis of variation is potentially implemented differently discrete versus continuous families. The 17 axes of variations are:

- **State Dimension** controls the number of possible states in discrete MDPs and the state vector dimension in continuous MDPs.
- **Action Dimension** controls the number of possible discrete actions in all families of MDPs.

- **Observation Dimension** controls the observation vector dimension in all families of MDPs.
- **Initial State Variance** controls the temperature of the distribution of the intial state in discrete MDPs and the standard deviation of the gaussian sampled from in continuous MDPs.
- **Transition Complexity** controls the number of layers in the NN which describes the transition function (for non discrete MDPs).
- **Transition Locality** controls how far delta transitions can go to from the previous state (for continuous MDPs).
- **Transition Stochasticity** controls the temperature of the softmax distribution of the next state distribution in discrete MDPs and the standard deviation of the Gaussian distribution in continuous MDPs.
- **Observation Complexity** controls the number of layers in the NN which describes the observation function (for non discrete MDPs).
- **Observation Stochasticity** controls the standard deviation of the Gaussian distribution centered at the observation mean.
- **Reward Complexity** controls the number of layers in the NN which describes the reward function (for non discrete MDPs).
- **Reward Stochasticity** controls the standard deviation of the Gaussian distribution centered at the reward mean.
- **Reward Sparse** is a boolean indicator variable which controls if the reward should be thresholded and converted to a 0-1 signal.
- **Reward Sparsity Probability** controls what percentage of states contain a reward when the reward is sparse.
- **Done Complexity** controls the number of layers in the NN which describes the done function (for non discrete MDPs).
- **Done Stochasticity** controls how stochastic the done signal is.
- **Done Sparsity Probability** controls what percentage of states are terminal done states.
- **Time Limit** controls the maximum episode length.

3.2.3 Random MLPs



(a) Before normalization (b) After normalization

Figure 2: Input-output mappings for random MLPs before and after normalization.

Some of the synthetic MDPs use randomly initialized (Multilayer Perceptron) MLP networks to get random function mappings. However, default weight initialization schemes don't result in networks

which have normally distributed outputs or strong correlations between inputs and outputs. We apply Batch Normalization [Ioffe and Szegedy, 2015] layers after every dense layer and initialize these layers once by forward passing standard normal inputs. The resulting input-output mappings before and after normalization are visualized in Figure 2. Adding multiple layers gives a natural way to increase the "complexity" of the input-output mapping.

3.3 Data Augmentations

To prevent memorization and induce ICL, we randomly project observations and actions using square orthogonal matrices (Kirsch et al. [2022b], Kirsch et al. [2023], Lu et al. [2024]). Importantly, these matrices are the same across the context of the transformer, allowing the transformer to in-context learn the specific task induced by a random projection.

3.4 Algorithm: In-Context Imitation Learning

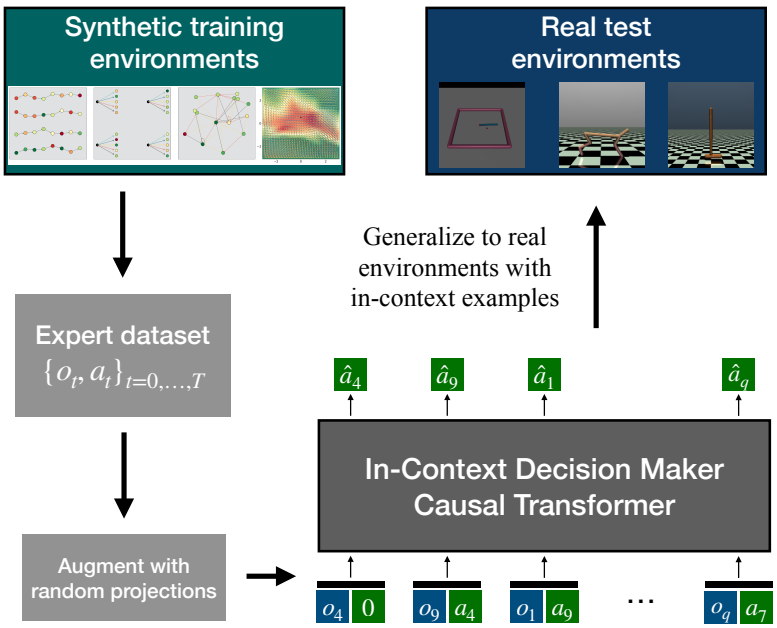


Figure 3: A summary of our approach. First, a distribution of synthetic MDPs is constructed. Next, an expert dataset is collected by training specialist PPO agents. Then, the dataset is augmented via random projections to create a large diversity of tasks. A transformer is then trained to predict the query action given a query observation and a context of observations and actions. Finally, at test time, we generalize to a completely new environment (different observations, actions, dynamics) purely in-context, given a handful of expert observation-action pairs.

For each environment, the expert dataset for that environment is in the format $\{(o_i, a_i)\}_{i=1,\ldots,N}$ where o_i is the observation and a_i is the action taken.

3.4.1 Training

During training, we sample T (the context length), samples from the dataset and augment them as described in Section 3.3. We feed these into a causal transformer model as shown in Figure 3. The transformer, at each token position, is trained to predict the action corresponding to that observation, given the previous observation and actions. With enough data diversity, driven by the augmentations, the transformer learns to utilize the in-context examples for prediction (Kirsch et al. [2022b]).

3.4.2 Inference/Evaluation

During inference on a new test environment, we assume we are given $T-1$ observations and action pairs from an expert to imitate. We use these observations and actions as the prompt to the model. Each new query observation gets appended to the prompt, at the final token position, and the transformer's forward pass produces the query's action. Note, there is no gradient fine-tuning when adapting to a new environment.

4 Experiments

4.1 Hyperparameters

Table 1: Hyperparameters used in the model training

Hyperparameter	Value
Input Observation Dimension	32
Output Action Dimension	8
Context Length	128
Number of Layers	4
Number of Attention Heads	8
Hidden Dimension	256
Batch Size	128
Learning Rate	3×10^{-4}
Learning Rate Schedule	constant
Weight Decay	0
Clip Gradient Norm	1
Number of Iterations	100000
Number of Augmentations	1000000

Our experiments focus on two research questions: (1) Can synthetic environments be as effective as real environments in producing in-context imitation learners for decision making? (2) What properties of the synthetic environment distributions affect generalization the most?

4.2 Effectiveness of synthetic environment distributions

To analyze the effectiveness of our synthetic environment distributions we train a variety of imitation learning agents on different expert policy datasets.

The *random baseline* corresponds to an untrained neural network. In terms of required real training data or environment interactions, training on synthetic data is just as cheap as not training at all, hence any return larger than the random baseline would be a 'free' gain from synthetic data. The *oracle augmented* corresponds to training on the test environment data augmented by random projections. In this setting, the agent is forced to make use of the expert demonstrations to perform in-context learning, but still has seen data from the test environment. Thus, this corresponds to an 'easy' in-distribution setting. We 0-1 normalize all returns according to these two settings. The *gato augmented* baseline is trained on all the other ten Mujoco environments (excluding the test environment) and is also augmented with random projections to improve generalization. This is the baseline our proposed synthetic environments are competing with.

From Figure 4 we find that training on purely synthetic environment data is competitive or better than training on real environments ("gato augmented") on a range of test environments. Interestingly, there does not seem to be much difference between the best MDP distributions across the different MDP families. There is still a significant gap between all methods and the oracle, which is expected, since we are testing the much more difficult out-of-distribution setting.

4.3 Properties of synthetic environment distributions

To analyze which properties of our synthetic environment distributions affect generalization the most, we vary these properties for all environment families and report the results in Figure 5. Because of

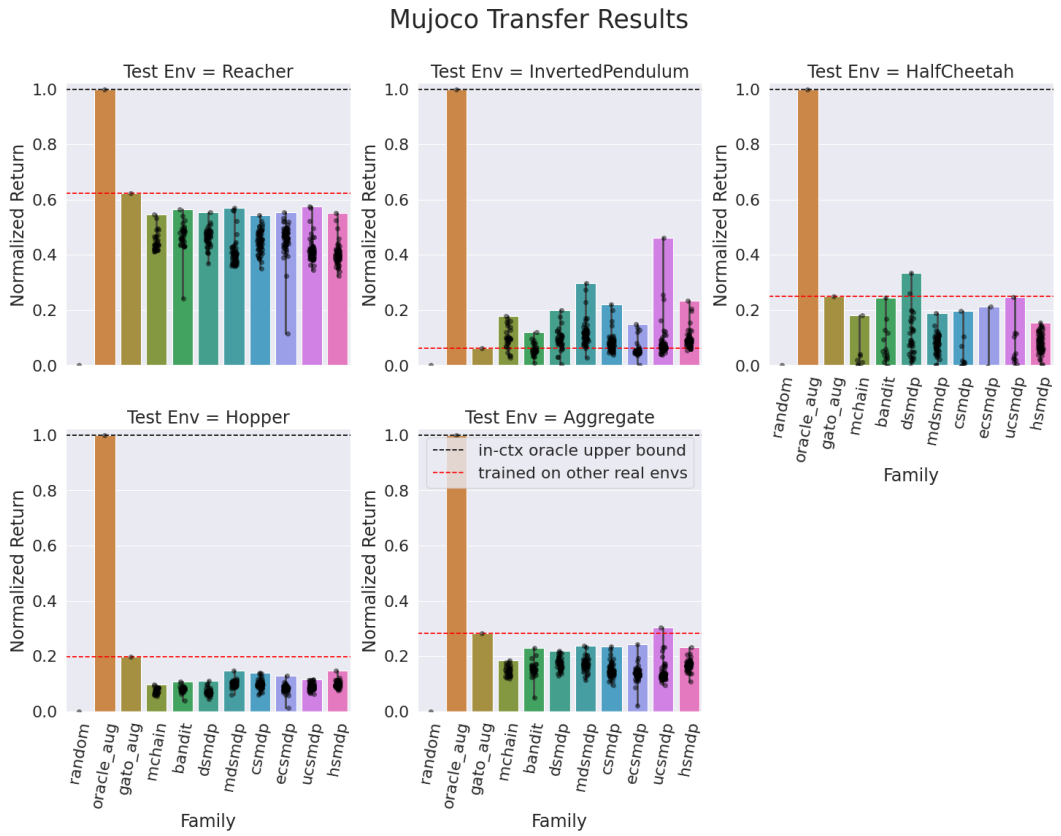


Figure 4: Transfer performance on real environments with different sources of pretraining data. "random" and "oracle_aug" are baselines used to normalize values and represent lower and upper bounds on performance. "gato_aug" is trained on other real environments and is the baseline to compare against. The others are our different families of synthetic MDPs. The bar represents the best environment distribution found within the family, while the black dots represent all environment distributions.

the diverse nature of the families, each axis of variation is binned into 5 values of (very small, small, medium, large, very large). When each axis is varied, the other axis values are set to the medium setting. Observation dimension, initial state variance, transition locality, and observation stochasticity seem to have the strongest correlations with performance. Other properties seem to not matter much. We hypothesize these properties would matter more if we evaluated in-context trajectory modeling rather than in-context imitation learning, because the context would contain temporally correlated observations and actions.

5 Discussion

This work evaluated in-context imitation learning capabilities when pre-trained on synthetic environments and evaluated on real environments. We find that synthetic environments are surprisingly competitive to real environments in this setting. One direction forward is to evaluate in-context meta-RL capabilities instead, eliminating the need for inference time expert examples [Kirsch et al., 2023]. Another interesting direction forward is to apply Unsupervised Environment Design (UED) [Wang et al., 2020a, Dennis et al., 2021, Parker-Holder et al., 2023] to the synthetic MDPs to create more task diversity specific to the agent's current weaknesses. Because some of the synthetic MDP families are modeled by neural networks, which are universal function approximators, UED might unlock an Open-Ended [Kenneth O. Stanley, 2017, Clune, 2020] process where the agent gradually masters the entire family of MDPs.

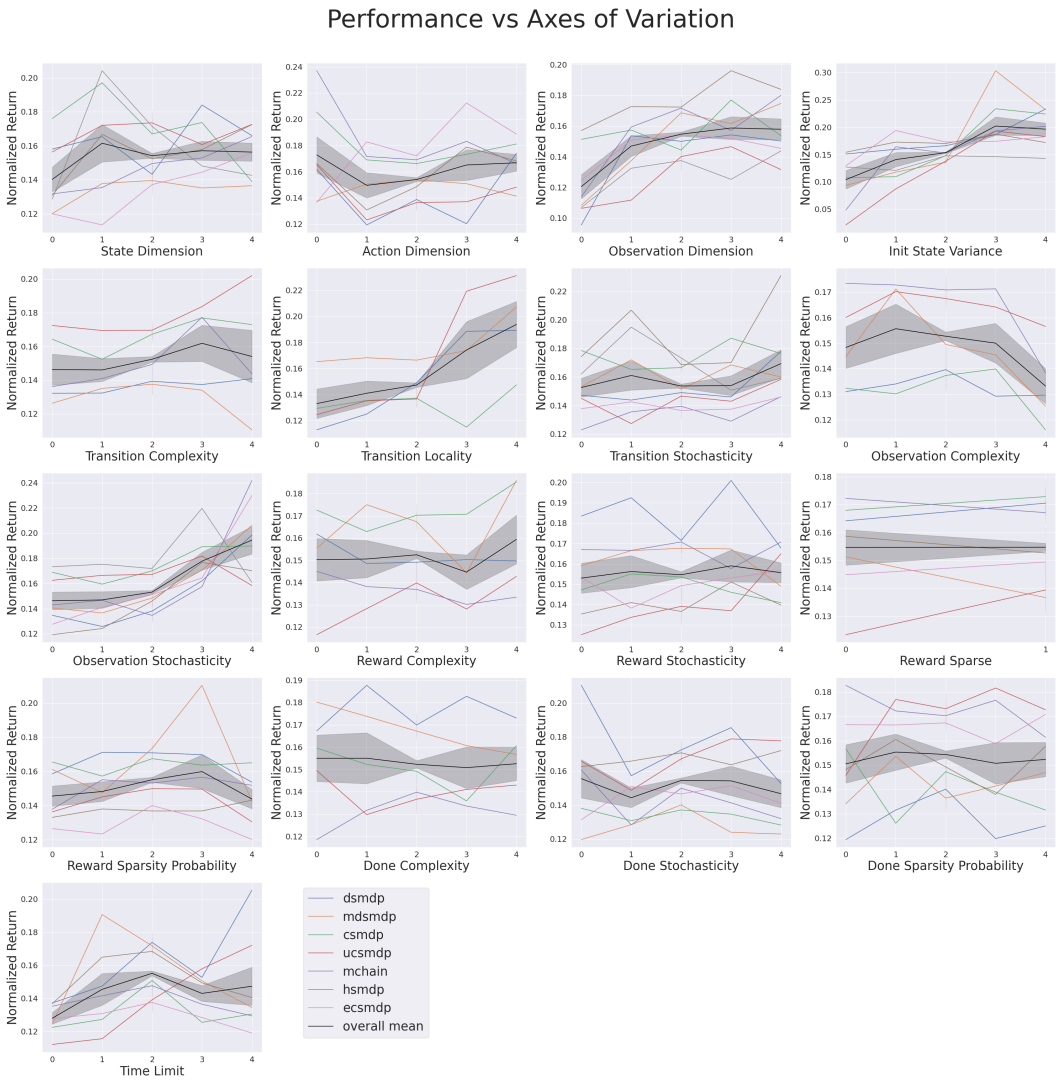


Figure 5: Transfer performance on real environments when varying pretraining environment properties. Each property is implemented differently for each family, so they are discretized to five values.

References

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise, 2022.

Manel Baradad, Chun-Fu Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. Procedural image programs for representation learning, 2023.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL `http://dx.doi.org/10.1613/jair.3912`.

Cheng-Han Chiang and Hung yi Lee. On the transferability of pre-trained language models: A study from artificial datasets, 2022.

Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence, 2020.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 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, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Fabio Ferreira, Thomas Nierhoff, and Frank Hutter. Learning synthetic environments for reinforcement learning with evolution strategies, 2021.

Fabio Ferreira, Thomas Nierhoff, Andreas Saelinger, and Frank Hutter. Learning synthetic environments and reward networks for reinforcement learning, 2022.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images, 2021.

Lisa Soros Kenneth O. Stanley, Joel Lehman. Open-endedness: The last grand challenge you've never heard of — oreilly.com. `https://www.oreilly.com/radar/open-endedness-the-last-grand-challenge-youve-never-heard-of/`, 2017.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, January 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14174. URL `http://dx.doi.org/10.1613/jair.1.14174`.

Louis Kirsch, Sjoerd van Steenkiste, and Juergen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. In *International Conference on Learning Representations*, 2020. 2019 arXiv preprint arXiv:1910.04098.

Louis Kirsch, Sebastian Flennerhag, Hado van Hasselt, Abram Friesen, Junhyuk Oh, and Yutian Chen. Introducing symmetries to black box meta reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7202–7210, 2022a. 2021 arXiv preprint arXiv:2109.10781.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022b.

Louis Kirsch, James Harrison, Daniel Freeman, Jascha Sohl-Dickstein, and Jürgen Schmidhuber. Towards general-purpose in-context learning agents. *Foundation Models for Decision Making Workshop at NeurIPS*, 2023.

Kundan Krishna, Jeffrey Bigham, and Zachary C. Lipton. Does pretraining for summarization require knowledge transfer?, 2021.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation, 2022.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-game decision transformers, 2022.

Jarek Luca Liesen, Chris Lu, Andrei Lupu, Jakob Nicolaus Foerster, Henning Sprekeler, and Robert Tjarko Lange. Discovering minimal reinforcement learning environments, 2024. URL `https://openreview.net/forum?id=VDkye4EKVe`.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Thomas Miconi. Procedural generation of meta-reinforcement learning tasks, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

Junhyuk Oh, Matteo Hessel, Wojciech M Czarnecki, Zhongwen Xu, Hado P van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, 33:1060–1070, 2020.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design, 2023.

Sharath Chandra Raparthy, Eric Hambro, Robert Kirk, Mikael Henaff, and Roberta Raileanu. Generalization to new sequential decision making tasks with in-context learning, 2023.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning?, 2022.

Ryokan Ri and Yoshimasa Tsuruoka. Pretraining with artificial language: Studying transferable knowledge in language models, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeff Clune, and Kenneth O. Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions, 2020a.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020b.

Zecheng Wang, Che Wang, Zixuan Dong, and Keith Ross. Pre-training with synthetic data helps offline reinforcement learning, 2024.