XLand-100B: A Large-Scale Multi-Task Dataset for In-Context Reinforcement Learning

Alexander Nikulin[*] AIRI MIPT
Ilya Zisman[*] AIRI Skoltech
Alexey Zemtsov[*] T-Bank
Vladislav Kurenkov AIRI Innopolis University

Abstract

Following the success of the in-context learning paradigm in large-scale language and computer vision models, the recently emerging field of in-context reinforcement learning is experiencing a rapid growth. However, its development has been held back by the lack of challenging benchmarks, as all the experiments have been carried out in simple environments and on small-scale datasets. We present **XLand-100B**, a large-scale dataset for in-context reinforcement learning based on the XLand-MiniGrid environment, as a first step to alleviate this problem. It contains complete learning histories for nearly $30,000$ different tasks, covering 100B transitions and 2.5B episodes. It took $50,000$ GPU hours to collect the dataset, which is beyond the reach of most academic labs. We also benchmark common in-context RL baselines and show that they struggle to generalize to novel and diverse tasks. With this substantial effort, we aim to democratize research in the rapidly growing field of in-context reinforcement learning and provide a solid foundation for further scaling.

1 Introduction

In-context learning, i.e. the ability to learn new tasks purely based on examples given in the context during inference and without any weight updates, was initially thought to be an emergent property of large language models, such as GPT-3 (Brown et al., 2020). However, it was quickly discovered that small transformers are also capable of in-context learning (Kirsch et al., 2022; Von Oswald et al., 2023), and even many non-transformer models have such abilities (Bhattamishra et al., 2023; Akyürek et al., 2024; Park et al., 2024; Grazzi et al., 2024; Vladymyrov et al., 2024; Tong & Pehlevan, 2024). More importantly, driven by properties of data, rather than the architecture (Chan et al., 2022; Gu et al., 2023), in-context learning is not specific to language modeling and has been found in other domains, e.g. image generation (Bai et al., 2023; Najdenkoska et al., 2023; Doveh et al., 2024; Tian et al., 2024).

However, despite the rapid adoption of the transformer architecture in reinforcement learning (RL) after the release of Decision Transformer (DT) (Chen et al., 2021; Hu et al., 2022; Agarwal et al., 2023; Li et al., 2023), models with in-context learning capabilities appeared only recently. This delay is caused by a number of reasons. Firstly, to transition from in-weights to in-context learning, a model should be trained on tens of thousands of unique tasks (Kirsch et al., 2022). Unfortunately, even the largest RL datasets currently contain only hundreds or tasks (Padalkar et al., 2023). Secondly, it was necessary to determine the right way to provide a context to a transformer and develop the data

[*]Equal contribution.
Correspondence to: nikulin@airi.net
Work is done at dunnolab.ai

Workshop on Adaptive Foundation Models at 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Table 1: Comparison with other RL datasets.

Data	# tasks	# transitions	# episodes	Size	Open-Source	Enables ICRL
XLand-100B (ours)	28,876	100B	2.5B	320GB	✓	✓
JAT	157	323M	N/A	1TB	✓	✗
GATO	596	1.5T	63M	N/A	✗	✗
Open X-Embodiment	527	N/A	2.4M	9TB	✓	✗
AlphaStar Unplugged	1	21B	2.8M	N/A	✗	?
NetHack	1	3.5B	110K	97 GB	✗	✗
D4RL	11	N/A	40 M	4.8GB	✓	✗
V-D4RL	4	2.4M	N/A	17GB	✓	✗
D5RL	50	N/A	2500	N/A	✗	✗
RL Unplugged	90	80M	N/A	54.1TB	✓	✗
Procgen	16	37M	N/A	500GB	✓	✗

collection pipeline, which for many methods (Laskin et al., 2022; Lee et al., 2023; Shi et al., 2024) was different from what is commonly available in existing datasets (see Appendix B).

Due to the lack of suitable datasets and the high cost of collecting data in existing environments, the recent wave of in-context RL research (Laskin et al., 2022; Lee et al., 2023; Norman & Clune, 2023; Kirsch et al., 2023; Sinii et al., 2023; Zisman et al., 2023) used environments with very simple task distributions, where it was feasible to collect datasets with hundreds of tasks. While these benchmarks are affordable, they are not suitable for the comparison of methods at scale on tasks with high diversity and difficulty, which is essential for real-world applications. Because of this, the development of in-context RL is currently hindered by these factors. We believe it is crucial to address these barriers, given the essential role of in-context learning in the path to foundation models and truly generalist agents (Team et al., 2021, 2023; Kirsch et al., 2023; Lu et al., 2024; Liu et al., 2024).

2 XLand-100B Dataset

We present **XLand-100B**, a large-scale dataset for in-context RL based on the XLand-MiniGrid (Nikulin et al., 2023) environment, together with its smaller and simpler version **XLand-Trivial-20B**. Combined they contain about 3.5B episodes, 130B transitions and 40,000 unique tasks (see Table 2 in the Appendix B for detailed statistics). It took $50,000$ GPU hours to collect the dataset, which is beyond the reach of most academic labs. In contrast to most existing datasets for RL, our dataset is compatible with the most widely used in-context learning RL methods. We provide additional background in Appendix A and compare our dataset in detail with the currently existing ones in Appendix B.

2.1 Data Format

Storage format. We chose to store the datasets in HDF5 file format based on its popularity and convenience. We used gzip compression with default compression strength of 6, which reduced dataset size from almost 5TB+ to just $\sim$600GB for our main dataset. Using a little trick described later, we were able to reduce the size even more to just 326 GB (see Table 2). However, a naive use of compression can dramatically increase batch sampling time and slow overall training time down. We tuned HDF5 cache chunk size specifically to maximize sampling throughput for large sequence lengths. After tuning, we achieved a fourfold speedup over the naive compression, and were only two times slower compared to no compression. See Appendix D for throughput benchmarks.

Data and metadata format. We collect complete learning histories, i.e. we store all observations, actions, rewards and dones encountered during agent training (see Appendix C for more details). To be compatible with DPT-like methods (Lee et al., 2023), we also store expert actions for each transition (see Section 2.2). Unlike popular formats such as RLDS, we store transitions as one sequential array per history per modality. The rationale here is that under compression, it is much cheaper to sample slices of long episodes in sequence rather than sampling across different groups.

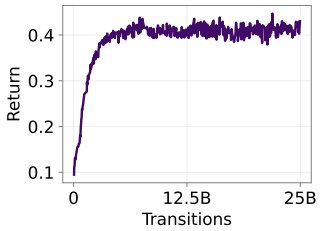

Figure 2: Evaluation return for multi-task goal-conditioned reccurent PPO pretraining on 65k tasks. Pretrained agent was further used a starting point for single-task finetuning during dataset collection.

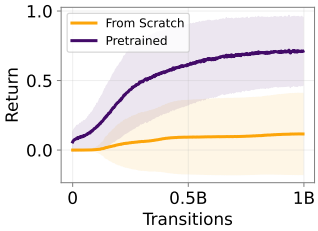

Figure 3: Single-task evaluation curves on 36 hard tasks for policies trained from scratch or fine-tuned from multi-task pretrained checkpoints. See Appendix E for curves on tasks of all difficulty.

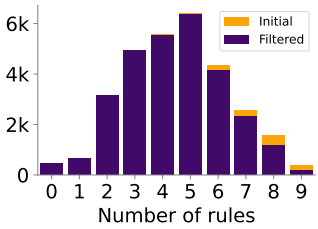

Figure 4: Distribution of the tasks by difficulty sampled initially and in the resulting dataset. To ensure the quality, we filtered tasks where the final return was below 0.3 or the data was corrupted due to some errors during training.

We also store observations efficiently to further reduce dataset size. Instead of storing two channels for tile and color, we map their indexes into Cartesian product of colors and tiles, halving the storage size. They can be decoded easily during sampling without any overhead with `divmod` function. In addition, for each history we store the XLand-MiniGrid environment ID, benchmark ID and ruleset ID, which can be used later to filter the dataset, e.g. based on the complexity of the tasks, split into train and test or set up the environment for evaluation.

2.2 Data Collection

At a high level, data collection was organised into three stages, namely multi-task task-conditioned pre-training; single-task fine-tuning to collect learning histories; and finally post-processing and filtering. Although we used highly optimised GPU-accelerated implementations of the base RL algorithm and environment, it still took $50,000$ GPU hours to collect the full dataset. Collecting a dataset of this size for any other environment suitable for in-context RL, e.g. Meta-World (Yu et al., 2020), would take much longer, which is unlikely to be feasible for most practitioners (Nikulin et al., 2023). Next, we describe the collection process, including the selection of the base RL algorithm and all subsequent steps. We provide exact hyperparameters for each stage in Appendix M.

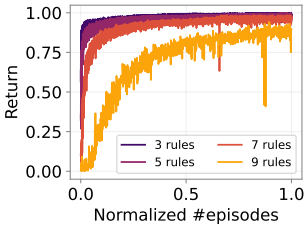

Figure 1: Learning histories for the XLand-100B dataset separated by number of rules. For visual clarity, we show only a sample of the possible number of rules and normalize the number of episodes, as they may vary considerably.

Base algorithm. For our datasets we chose PPO (Schulman et al., 2017), due its high scalability and compatibility with massively parallel environments. We ported the implementation from recurrent PPO provided by (Nikulin et al., 2023), customizing it to meet our needs. We added callbacks for saving transitions during training and extended agent architecture to take ruleset encoding as an optional condition for pre-training. We used GRU (Cho et al., 2014) for memory, as it showed satisfactory performance on preliminary experiments. Since the base algorithm was implemented in JAX (Bradbury et al., 2018), we were able to just-in-time compile the entire training loop, achieving 1M steps per second during training on one GPU with mixed precision enabled.

Pretraining. For our main XLand-100B dataset we uniformly sampled tasks from `medium-1m` benchmark from XLand-MiniGrid. It contains tasks of various difficulty, ranging from zero to nine rules. Unfortunately, on many hard tasks our base algorithm could not manage to converge in the time budget allocated for a single training run. On the hardest tasks, it was not even possible to get a non-zero reward at all, due to the exploration challenge that such tasks poses. In order to speed up convergence and exploration on harder tasks, we pre-train an agent in a multi-task task-conditioned manner. We expose the ruleset specification, which is usually hidden from the agent, and encode the goal and rules via embeddings, concatenating resulting encodings and passing it as an additional input to the agent. After that, we train the agent on 65k tasks simultaneously for 25B transitions. As Figure 2 shows, such an agent learns to generalize zero-shot on new tasks quite well, although we do

not aim to push it to the limit, as zero-shot generalization does not produce a smooth learning history during fine-tuning. We skip this stage for XLand-Trivial-20B dataset due to the simplicity of the tasks in `trivial-1m` benchmark.

Finetuning. This is a key stage in the data collection process, during which we finetune a pretrained agent while recording the transitions encountered into the dataset. We finetune the agent using 8192 parallel environments for 1B transitions on 30k uniformly sampled tasks from `medium-1m` benchmark. We mask out the task-conditioning encoding to prevent zero-shot generalization. We record transitions only from first 32 parallel environments. For the XLand-Trivial-20B dataset, instead of fine-tuning, we train the agent from scratch on 10k uniformly sampled tasks from the `trivial-1m` benchmark, keeping all other hyperparameters the same. In the Figure 3 we show the effect of finetuning on hard tasks (with more than seven rules) compared to training from scratch. It can be seen that we are able to show strong performance even on the hardest tasks, increasing the diversity and coverage of the resulting dataset. For the same results on tasks of all levels of difficulty, see Appendix E.

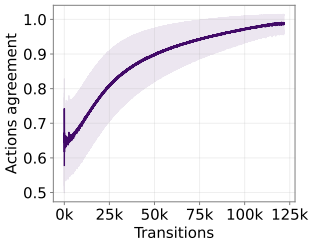

Figure 5: Agreement between actions predicted by the expert and the actual actions in the learning history. We use final PPO policy as an expert for actions labeling.

Postprocessing. After fine-tuning, it was necessary to additionally label the transitions with the expert actions to support DPT-like methods (Lee et al., 2023). To do this, we walk through the entire learning history with the final policy, starting from the initial hidden state for the GRU. We evaluate the validity of such a labelling scheme later in the Section 2.3. To ensure quality, we filtered out any task with a final return below 0.3 as an unrepresentative learning history. There were some failures, such as GPU crashes, which are inevitable during large-scale training. So any runs with corrupted data were also filtered out. In total, we filtered out about 1k tasks, leaving almost 29k tasks in the final dataset. We provide detailed statistics for each dataset in Table 2 and the final distribution of tasks by number of rules in Figure 4.

2.3 Data Evaluation

In this section we validate that the resulting XLand-100B dataset actually fulfils the two most important requirements for in-context RL, namely it contains learning histories with distinct policy improvement pattern and has expert actions for each transition (see Appendix B for a discussion). We provide analogous results for XLand-Trivial-20B in the Appendix F.

Improvement history. In the Figure 1 we show the averaged return from the learning histories separated by the number of rules. In order to better show the speed of learning on the same scale, we have normalized the x-axis for each level of difficulty, as the number of episodes can vary greatly (as it takes more time to solve complex tasks). One can see that the dataset provides a whole range of learning speeds, from very fast on easy problems to much slower on the hardest, which may be important for methods based on AD (Zisman et al., 2023; Shi et al., 2024). For a learning history averaged over full dataset see Appendix F.

Expert actions relabeling. In contrast to AD, which predicts next actions from the trajectory itself, DPT-like methods require access to optimal actions on each transition for prediction. However, for the most nontrivial or real-world problems, obtaining true optimal actions in large numbers is unlikely to be possible. Recently, Lin et al. (2023) introduced approximate DPT, scheme where expert actions are estimated from the entire history by some algorithm. We implemented this scheme for lack of evident alternatives. However, we had to make sure that such a labeling is adequate in our case and the expert at the end predicts actions close to what the policy did in reality near the end of training. This is not obvious, as during the labeling it can diverge into out-of-distribution hidden states for GRU. In the Figure 5 we show that on XLand-100B the agreement between the predicted actions by the expert and the actual actions increases closer to the end of the learning history, meaning that the expert does not diverge during relabeling.

3 Experiments

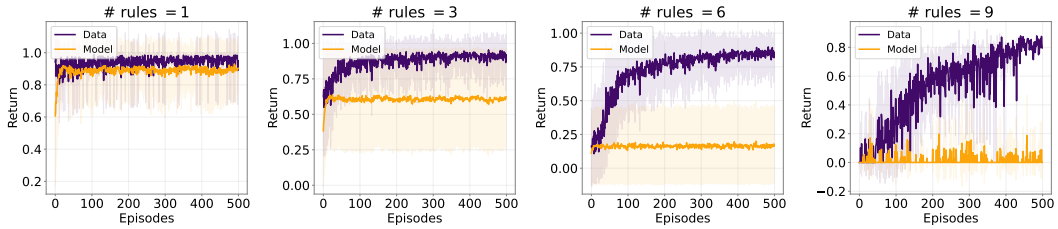

Figure 6: Comparison of the learning histories in `-100B` dataset vs. AD performance on the same *training* tasks. AD is able to solve simple tasks, however its performance degrades as the rulesets get deeper. The context length of the model is 1024. The evaluation parameters, except training tasks, are the same as in Figure 7.

In this section, we investigate whether our datasets can enable an in-context RL ability. Additionally, we demonstrate how well current in-context algorithms perform across different task complexities and outline their current limitations. We take AD (Laskin et al., 2022) and DPT (Lee et al., 2023) for our experiments, the exact implementations details are in Appendix H and Appendix I. Both methods were trained on `XLand-Trivial-20B` and `XLand-100B` with $\{512, 1024, 2048, 4096\}$ and $\{1024, 2048, 4096\}$ context lengths respectively. For evaluation, we run three models on 1024 unseen tasks for 500 episodes.

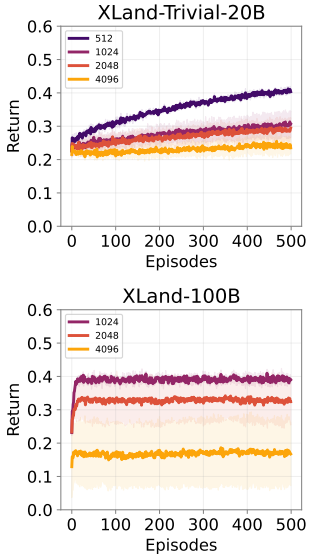

Figure 7: AD performance on our datasets for different sequence lengths. Both datasets lead to the emergence of in-context ability. We report the average return on 1024 unseen tasks across 3 seeds.

AD. AD shows an emergence of in-context ability during training on both datasets. Figure 7 demonstrates the performance of the method for different context lengths. On `-Trivial-20B` dataset it is able to show a stable policy improvement from about 0.28 to 0.4 during the evaluation. For `-100B` the performance is similar, but the pace of improvement is faster. We hypothesise that it happens due to wider data-coverage, since the agent sees more complex tasks and is able to learn faster from them. To further examine the performance on the `-100B` dataset, we evaluate AD on the training tasks from the dataset and separate the performance based on the complexity of the tasks. As shown in Figure 6, AD is able to demonstrate in-context abilities on simple tasks, but it struggles with more complex ones. We believe there is a need for further research to discover new and more sample-efficient architectures capable of solving the more complex tasks of our dataset.

DPT. Decision-Pretrained Transformer (Lee et al., 2023) is another method that exhibit in-context RL capabilities. However, in our experiments we were unable to train it so that these abilities emerge (see Appendix L). We believe this is closely connected to the inability of DPT to reason in POMDP environments. For a detailed investigation, we refer the reader to Appendix J.

4 Limitations and Future Work

There are several limitations to our work, some of which we hope to address in future releases. Despite the size and diversity of the datasets provided in terms of tasks, we do not provide diversity in terms of the domains, as all tasks share the same observation and action spaces. In addition, the tasks also share an overall latent structure, i.e. it is always a form of binary tree. This can be addressed with more diverse benchmark generators in the XLand-MiniGrid library (Nikulin et al., 2023). All learning histories were collected on grids with only one room, which may limit the transfer to the harder layouts with multiple rooms containing doors. Finally, the effect of fine-tuning from pre-trained checkpoints is underexplored and could potentially hurt performance, as there are many learning histories that start from a high reward. We hope to improve the RL baseline for data collection to avoid the need for multi-task pre-training in the future.

References

Agarwal, P., Rahman, A. A., St-Charles, P.-L., Prince, S. J., and Kahou, S. E. Transformers in reinforcement learning: a survey. *arXiv preprint arXiv:2307.05979*, 2023.

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024.

Bai, Y., Geng, X., Mangalam, K., Bar, A., Yuille, A., Darrell, T., Malik, J., and Efros, A. A. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

Beck, J., Vuorio, R., Xiong, Z., and Whiteson, S. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Bhattamishra, S., Patel, A., Blunsom, P., and Kanade, V. Understanding in-context learning in transformers and llms by learning to learn discrete functions. *arXiv preprint arXiv:2310.03016*, 2023.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chan, S., Santoro, A., Lampinen, A., Wang, J., Singh, A., Richemond, P., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Doveh, S., Perek, S., Mirza, M. J., Alfassy, A., Arbelle, A., Ullman, S., and Karlinsky, L. Towards multimodal in-context learning for vision & language models. *arXiv preprint arXiv:2403.12736*, 2024.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Gallouédec, Q., Beeching, E., Romac, C., and Dellandréa, E. Jack of All Trades, Master of Some, a Multi-Purpose Transformer Agent. *arXiv preprint arXiv:2402.09844*, 2024. URL https://arxiv.org/abs/2402.09844.

Grazzi, R., Siems, J., Schrodi, S., Brox, T., and Hutter, F. Is mamba capable of in-context learning? *arXiv preprint arXiv:2402.03170*, 2024.

Grigsby, J., Fan, L., and Zhu, Y. Amago: Scalable in-context reinforcement learning for adaptive agents. *arXiv preprint arXiv:2310.09971*, 2023.

Gu, Y., Dong, L., Wei, F., and Huang, M. Pre-training to learn in context. *arXiv preprint arXiv:2305.09137*, 2023.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., Dulac-Arnold, G., Li, J., Norouzi, M., Hoffman, M., Nachum, O., Tucker, G., Heess, N., and de Freitas, N. Rl unplugged: A suite of benchmarks for offline reinforcement learning, 2021.

Hambro, E., Raileanu, R., Rothermel, D., Mella, V., Rocktäschel, T., Küttler, H., and Murray, N. Dungeons and data: A large-scale nethack dataset, 2023.

Hu, S., Shen, L., Zhang, Y., Chen, Y., and Tao, D. On transforming reinforcement learning by transformer: The development trajectory. *arXiv preprint arXiv:2212.14164*, 2022.

Hui, D. Y.-T., Chevalier-Boisvert, M., Bahdanau, D., and Bengio, Y. Babyai 1.1, 2020.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Kirsch, L., Harrison, J., Freeman, C., Sohl-Dickstein, J., and Schmidhuber, J. Towards general-purpose in-context learning agents. In *NeurIPS 2023 Workshop on Generalization in Planning*, 2023. URL https://openreview.net/forum?id=eDZJTdUsfe.

Kurenkov, V., Nikulin, A., Tarasov, D., and Kolesnikov, S. Katakomba: Tools and benchmarks for data-driven nethack. *Advances in Neural Information Processing Systems*, 36, 2024.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., and Ye, D. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.

Lin, L., Bai, Y., and Mei, S. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *arXiv preprint arXiv:2310.08566*, 2023.

Liu, H. and Abbeel, P. Emergent agentic transformer from chain of hindsight experience. In *International Conference on Machine Learning*, pp. 21362–21374. PMLR, 2023.

Liu, X., Lou, X., Jiao, J., and Zhang, J. Position: Foundation agents as the paradigm shift for decision making. *arXiv preprint arXiv:2405.17009*, 2024.

Lu, C., Ball, P. J., Rudner, T. G. J., Parker-Holder, J., Osborne, M. A., and Teh, Y. W. Challenges and opportunities in offline reinforcement learning from visual observations, 2023.

Lu, C., Schroecker, Y., Gu, A., Parisotto, E., Foerster, J., Singh, S., and Behbahani, F. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Mathieu, M., Ozair, S., Srinivasan, S., Gulcehre, C., Zhang, S., Jiang, R., Paine, T. L., Powell, R., Żołna, K., Schrittwieser, J., Choi, D., Georgiev, P., Toyama, D., Huang, A., Ring, R., Babuschkin, I., Ewalds, T., Bordbar, M., Henderson, S., Colmenarejo, S. G., van den Oord, A., Czarnecki, W. M., de Freitas, N., and Vinyals, O. Alphastar unplugged: Large-scale offline reinforcement learning, 2023.

Mediratta, I., You, Q., Jiang, M., and Raileanu, R. The generalization gap in offline reinforcement learning, 2024.

Melo, L. C. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pp. 15340–15359. PMLR, 2022.

Mirchandani, S., Xia, F., Florence, P., Ichter, B., Driess, D., Arenas, M. G., Rao, K., Sadigh, D., and Zeng, A. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Najdenkoska, I., Sinha, A., Dubey, A., Mahajan, D., Ramanathan, V., and Radenovic, F. Context diffusion: In-context aware image generation. *arXiv preprint arXiv:2312.03584*, 2023.

Nikulin, A., Kurenkov, V., Zisman, I., Agarkov, A., Sinii, V., and Kolesnikov, S. Xland-minigrid: Scalable meta-reinforcement learning environments in jax. *arXiv preprint arXiv:2312.12044*, 2023.

Norman, B. and Clune, J. First-explore, then exploit: Meta-learning intelligent exploration. *arXiv preprint arXiv:2307.02276*, 2023.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Park, J., Park, J., Xiong, Z., Lee, N., Cho, J., Oymak, S., Lee, K., and Papailiopoulos, D. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

Rafailov, R., Hatch, K. B., Singh, A., Kumar, A., Smith, L., Kostrikov, I., Hansen-Estruch, P., Kolev, V., Ball, P. J., Wu, J., et al. D5rl: Diverse datasets for data-driven deep reinforcement learning. 2023.

Raparthy, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. Generalization to new sequential decision making tasks with in-context learning. *arXiv preprint arXiv:2312.03801*, 2023.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent, 2022.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shala, G., Biedenkapp, A., and Grabocka, J. Hierarchical transformers are efficient meta-reinforcement learners. *arXiv preprint arXiv:2402.06402*, 2024.

Shi, L. X., Jiang, Y., Grigsby, J., Fan, L., and Zhu, Y. Cross-episodic curriculum for transformer agents. *Advances in Neural Information Processing Systems*, 36, 2024.

Sinii, V., Nikulin, A., Kurenkov, V., Zisman, I., and Kolesnikov, S. In-context reinforcement learning for variable action spaces. *arXiv preprint arXiv:2312.13327*, 2023.

Team, A. A., Bauer, J., Baumli, K., Baveja, S., Behbahani, F., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024.

Tong, W. L. and Pehlevan, C. Mlps learn in-context. *arXiv preprint arXiv:2405.15618*, 2024.

Vladymyrov, M., von Oswald, J., Sandler, M., and Ge, R. Linear transformers are versatile in-context learners. *arXiv preprint arXiv:2402.14180*, 2024.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Wang, J., Blaser, E., Daneshmand, H., and Zhang, S. Transformers learn temporal difference methods for in-context reinforcement learning. *arXiv preprint arXiv:2405.13861*, 2024.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

Zisman, I., Kurenkov, V., Nikulin, A., Sinii, V., and Kolesnikov, S. Emergence of in-context reinforcement learning from noise distillation. *arXiv preprint arXiv:2312.12275*, 2023.

A Background

A.1 In-Context Reinforcement Learning

Multiple methods for in-context RL have come out, each offering a different way of training and organising the context (Laskin et al., 2022; Lee et al., 2023; Mirchandani et al., 2023; Liu & Abbeel, 2023; Raparthy et al., 2023; Shi et al., 2024). We focus on Algorithm Distillation (AD) (Laskin et al., 2022) and Decision-Pretrained Transformer (DPT) (Lee et al., 2023), which we chose as the main methods for our work due to their simplicity and generality.

Algorithm Distillation. AD (Laskin et al., 2022) was one of the first to show that in-context learning was possible in RL, and captures the details of many other more recent methods (Mirchandani et al., 2023; Liu & Abbeel, 2023; Shi et al., 2024) while remaining very simple. It trains a transformer, or any other sequence model, to autoregressively predict next actions given the history of previous interactions, i.e. observations, actions and rewards. To transition from in-weights to in-context learning, it is essential that the **context should contain multiple episodes ordered by an increasing return**, which is different from the way it is done in DT-like methods (Chen et al., 2021; Janner et al., 2021; Lee et al., 2022).

Decision-Pretrained Transformer. DPT is an alternative approach inspired by the Bayesian inference approximation (Müller et al., 2021). Unlike AD, it trains a transformer to **predict the optimal action for a query state given a random, task specific, context**. That is, the context that does not have to be ordered, but only has to contain transitions belonging to the same task. Thus, DPT requires access to optimal actions, but does not require a dataset of learning histories.

In addition, the theoretical analyses of AD and DPT methods (Lin et al., 2023; Wang et al., 2024) showed that they can implement near-optimal online RL algorithms such as Lin-UCB, Thompson sampling or even temporal difference (TD) methods solely during the forward pass.

A.2 XLand-MiniGrid

Starting from the seminal work of Wang et al. (2016); Duan et al. (2016); Finn et al. (2017) on meta-RL, much of the subsequent work (Zintgraf et al., 2019; Melo, 2022; Grigsby et al., 2023; Lu et al., 2024; Shala et al., 2024; Beck et al., 2024) has focused on environments that either have very simple task distributions, or have very small and limited distributions of hard tasks. This is because, to generalize in meta-RL, training needs to be performed on many different tasks, significantly increasing the cost and time required for experimentation. Recently, Nikulin et al. (2023) released XLand-MiniGrid, a GPU-accelerated environment and million-task benchmarks that significantly lowered the entry barrier for meta-RL research. We will describe it shortly here.

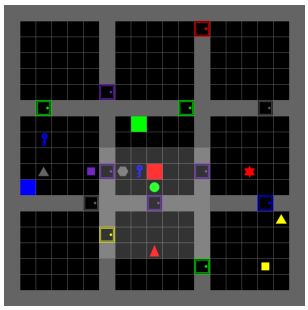

Figure 8: Visualization of a generic XLand-MiniGrid environment. Grid layout should be selected in advance, while the positions of the objects are randomized on each reset.

Environment. XLand-MiniGrid is a complete rewrite of MiniGrid (Chevalier-Boisvert et al., 2023) in JAX (Bradbury et al., 2018), incorporating a notion of rules and goals from XLand (Team et al., 2023). Leveraging JAX, it can run on a GPU or TPU accelerators at millions of steps per seconds. At its core, it is a goal-oriented grid-world environment with simple underlying dynamics, partial observability and sparse rewards. The action space is simple, consisting mainly of navigation and interaction with game objects, such as opening a door or picking and placing items. Observations are symbolic "images" encoding the agent surrounding as tile and color ID's. Rules are functions that can change the state of the environment based on some conditions, e.g. when two specific objects are places near each other, they both disappear and one new object is placed. Goals are similar, except they only validate some predefined conditions and do not change anything. Composing different rules and goals together we can create new tasks with varying reward and dynamics functions. For more detailed description we refer to Nikulin et al. (2023).

Benchmarks. Along with the environment itself, Nikulin et al. (2023) released a tool for the procedural generation of a vast number of unique tasks with varying levels of difficulty. Each task is represented by a binary tree, where the root is the goal to be achieved and rest of nodes define

rules of the environment to be triggered in a recurring sequence. To standardize comparisons, four pre-sampled benchmarks with increasing diversity were provided: `trivial`, `small`, `medium`, `high`, each with one million unique tasks. For this work, we chose `medium` as a middle ground between yet unsolved `high` and less challenging `small` benchmarks. We also use `trivial` for smaller and simpler dataset version (see Section 2).

B The Missing Piece For In-Context RL

In order to successfully train an in-context agent, training data shall meet certain criteria. To start with, the data should be comprised of actual learning histories (Laskin et al., 2022) or their approximations (Zisman et al., 2023), that contain enough exploration and exploitation phases of learning. Learning with just expert trajectories would not be sufficient for in-context ability to emerge, since an agent needs to know the history of policy improvement (Laskin et al., 2022; Kirsch et al., 2023). Another approach is to learn from optimal actions as proposed by Lee et al. (2023), but it is unclear how to access the optimal policies to get them. Besides, the data needs to contain thousands of different tasks to learn from (Kirsch et al., 2022). That is, for a simple task to find two squares on a 9×9 grid, an agent needs to see around 2000 different combinations of goals to start adapting for unseen locations (Laskin et al., 2022). Since such data was never collected and put into a single dataset, all current in-context RL practitioners were forced to generate data on their own, which inevitably added more complications in reproduction of the methods.

Besides, collecting thousands of different in-context episodes requires a considerable commitment, as training numerous RL agents is expensive in terms of time and resources. To that matter, the data used in current research is collected in very simplistic environments with straightforward goals, like reaching a specific target on a map (Laskin et al., 2022; Lee et al., 2023; Zisman et al., 2023) or to apply forces to actuators in order to walk a robot (Kirsch et al., 2023). This significantly slows down the pace of in-context RL research, as it is not only hard to test the applicability of proposed methods, but also yet unfeasible to determine the scaling laws in these environments.

To provide a complete picture for the reader, we briefly discuss the existing datasets and highlight why they are unfit for training in-context RL agents. For simplicity, we categorize them into two groups: classical datasets designed for offline-RL and datasets collected for large-scale supervised learning. Note that this categorization is fuzzy in nature and serves only for better understanding of the current structure in RL data.

Offline RL Datasets. The datasets in this category can be considered classical, as some of them exist for more than four years now (Fu et al., 2020). They were initially proposed for offline RL, containing simple tasks with a flat structure, e.g. perform locomotion with different robots (Lu et al., 2023) or path finding in a maze. Some of them also contain data from robotic manipulators (Fu et al., 2020; Rafailov et al., 2023), or even Atari frames (Gulcehre et al., 2021). Other datasets collect data for more sophisticated environments, such as NetHack Learning Environment (Hambro et al., 2023; Kurenkov et al., 2024) or ProcGen (Cobbe et al., 2019; Mediratta et al., 2024). However, the aforementioned datasets offer < 100 different tasks with a fixed policy (except for the `-replay` datasets, which have limited coverage of various policies). This limitation makes it difficult for in-context RL to emerge from such data. To overcome this pitfall, we collected almost $30,000$ tasks with a deep ruleset structure, that are a challenging problem to solve.

Large-Scale Supervised Pretraining. Recent progress in generalist agents, which can solve a multitude of environments, has been made possible thanks to large datasets. GATO dataset (Reed et al., 2022), however not being released to the public, consists of 1.5 trillion transitions along with 596 tasks, which makes it one of the largest dataset in RL. The open-sourced analogue, the JAT dataset (Gallouédec et al., 2024), is smaller in size with 157 tasks and 300 million transitions, but it provides comparable performance on most of the benchmarks. Both datasets contain expert RL demonstrations from BabyAI (Hui et al., 2020), Atari games (Bellemare et al., 2013), Meta-World (Yu et al., 2020) and more.

Another large dataset, Open X-Embodiment (Padalkar et al., 2023), is a combination of more than 60 datasets from different robotics research labs. It consists of 527 different tasks in robotics with the demonstrations from mostly from human experts. Despite the large quantity of transitions in these datasets, they do not contain learning histories with improving policies, making their application for

Table 2: Descriptive statistics of XLand datasets.

Dataset	**XLand-Trivial-20B**	**XLand-100B**
Episodes	868,805,556	2,500,152,898
Transitions	19,496,960,000	112,598,843,392
History length	60,928	121,856
Num tasks	10,000	28,876
Max task rules	0	9
Observation shape	(5, 5)	(5, 5)
Num actions	6	6
Mean final return	0.915	0.894
Median final return	0.948	0.925
Median episode transitions	22.45	57.75
Disk size (compressed)	60 GB	326 GB

in-context RL quite challenging. On the contrary, our XLand-100B dataset consists of 100 billions of transitions of RL agents' learning histories, making it possible for in-context abilities to emerge.

The only potentially applicable dataset to use for in-context RL is AlphaStar Unplugged (Mathieu et al., 2023). Although the authors did not initially plan to collect a suitable dataset, the data can be sorted by players' MMR (analogous to Elo rating). This sorting can be considered a steady policy improvement, thus enabling the in-context RL ability. For more details on the datasets, refer to Table 1.

C What is Inside Dataset?

Both `-Trivial` and `-100B` dataset are HDF5 files holding the same structure. The dataset is grouped the following way:

```
data["{key_id}/{entity_name}"][learning_history_id]
```

where `key_id` is an ordinal number of a task in dataset, `learning_history_id` is a learning history number from 0 to 32 and `entity_name` is one of the names mentioned in Table 3.

NB! Do not confuse `key_id` with the task ID, which should be accessed via

```
data["{key_id}"].attrs["ruleset-id"]
```

Table 3: Data description

Name	**Type**	**Shape**	**Description**
`states`	`np.uint8`	(5, 5)	s_t, colors and tiles from agent's POV
`actions`	`np.uint8`	scalar	a_t, from 0 to `NUM_ACTIONS`
`rewards`	`np.float16`	scalar	r_t, which agent recieved at timestep t
`dones`	`np.bool`	scalar	d_t, terminated or truncated episode flag
`expert_actions`	`np.uint8`	scalar	same as a_t but from a generating policy

D Compression Chunk Size Tuning

Table 4: Throughput with PyTorch dataloader with different HDF5 compression chunk size settings. We used 2048 sequence length, 64 batch size and 8 workers. Chunking was applied along the learning history dimension.

Compression	Chunk size	Throughput
None	None	1,549,619
gzip	None	173,895
gzip	256	423,513
gzip	512	549,397
gzip	1024	666,152
gzip	2048	768,706
gzip	4096	749,851
gzip	8192	737,646

E Additional Figures of Data Collection

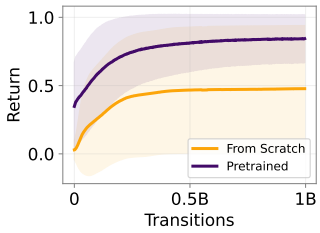

Figure 9: Single-task evaluation curves on 256 tasks for policies trained from scratch or fine-tuned from multi-task pretrained checkpoints.

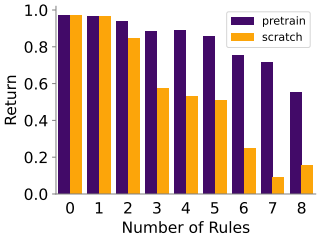

Figure 10: Final return by number of rules on 256 tasks for policies trainef from scratch or fine-tuned from multi-taks pretrained checkpoints.

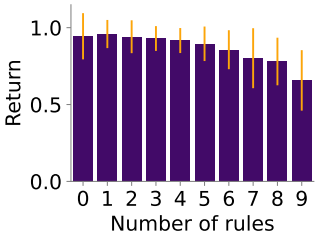

Figure 11: Final return by number of rules in the final XLand-100B dataset after post-processing.

F Additional Figures of Data Evaluation

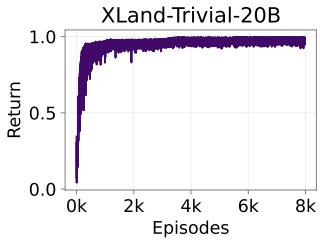

Figure 12: Return averaged over all learning histories in the final XLand-Trivial-20B dataset.

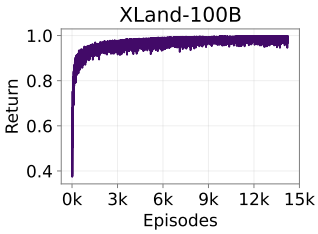

Figure 13: Return averaged over all learning histories in the final XLand-100B dataset.

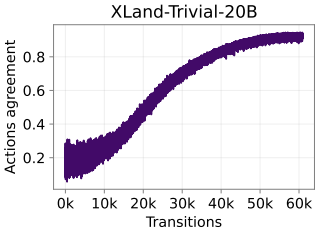

Figure 14: Agreement between actions predicted by the expert and the actual actions in the learning history. We use final PPO policy as an expert for actions labeling.

G Details of In-Context Evaluation

The evaluation process resembles the standard in-weight learning, with the key difference being that the learning itself happens during evaluation. When interacting with the environment, the agent populates the Transformer's context with the latest observations, actions and rewards. At the start of the evaluation, the context is empty. Note that the agent's context is cross-episodic, which makes it possible for the agent to access transitions in previous episodes. We report the cumulative reward that the agent achieved at the end of each episode. We run evaluation for each model for 500 episodes, reporting mean return across 1024 unseen tasks with standard deviation across 3 seeds. We claim that the in-context emerged when the mean return rises up until some level. This means the agent improves its policy from episode to episode, learning how to solve a task.

H AD Implementation

Table 5: Time for training and evaluation.

sequence length	1024	2048	4096
train	15 hrs	10 hrs	11.5 hrs
eval	20 min	30 min	50 min

We implement AD following the original paper of Laskin et al. (2022). To optimize the speed of training and inference we use FlashAttention-2 (Dao, 2024) with KV-caching and ALiBi positional embeddings (Press et al., 2022). We also concatenate observations, actions and rewards along the embedding dimensions, as it reduces the context size by the factor of three. We also use DeepSpeed (Rasley et al., 2020) to enable distributed training. The total number of parameters of our model is 25 M.

The approximate time of training for single epoch on a `-100B` dataset and evaluation on 1024 tasks on 8 H100 GPUs is shown in the Table 5. The computations were done on an internal cluster.

The hyperparameters was copied from (Laskin et al., 2022) except for the size of the network, it was scaled up to 25 M. The exact hyperparameters can be found in Table 9.

We also show training logs for both datasets. Trivial: wandb; medium: wandb

Table 6: Last episode returns for for AD algorithm on our datasets.

Dataset	Seq. length			
	512	**1024**	**2048**	**4096**
trivial-20B	0.403	0.306	0.294	0.234
medium-100B	0.416	0.400	0.324	0.200

I DPT Implementation

Our implementation is based on the original one (Lee et al., 2023). Compared to AD implementation (Laskin et al., 2022), during training phase, the model context is generated with decorrelated dataset transitions to increase data diversity and model robustness: given a query observation and a respective expert action for it, an in-context dataset is provided by random interactions within the same ruleset. During evaluation phase, multi-episodic contextual buffer consists only of previous episodes and updates after the current one ends. The intuition behind this approach is the observed policy during any given episode is fixed, so it is a lot easier to analyze this policy than a dynamically changing one while it is executing.

Both AD and DPT shares the same observation encoder and transformer block, except there is no positional encoding in DPT, as stated in (Lee et al., 2023).

The training consisted of 3 epochs due to computational and time limitations as 1 epoch approximately lasted 12 hours, while the evaluation on 1024 rulesets on 500 episodes could take from 5 to 21 hours,

depending on the model's sequence length. All experiments ran on 8 A100 GPUs. The computations were done on an internal cluster.

The hyperparameters was copied from (Lee et al., 2023) except for the size of the network, it was made up to 25 M, and sequence length, it was increased due to complexity of the tasks. The exact hyperparameters can be found in Table 8.

We also show DPT training logs. Trivial: wandb; medium: wandb

J On Limitations of DPT in POMDP

We additionally demonstrate the inability of DPT to learn in-context in Partially Observable MDP (POMDP) on the example of a toy Dark Key-To-Door environment (Laskin et al., 2022). The agent is required to find an invisible key and then open an invisible door. The reward of 1 is given when the agent first reaches the key and then the door. Note that the door cannot be reached until the key is found. This way Key-To-Door can be considered a POMDP. However, the environment can be reformulated as an MDP by providing additional boolean indicator of reaching a key in addition to the agent's position. This way, algorithms that work only with MDPs are able to solve this environment.

Based on this fact, we learn two different Q-tables for both environments: with and without the key indicator. The learning histories of Q-Learning algorithm are stored together with optimal actions computed via the oracle.

For clarity, we call DPT training and evaluation Markovian when "reached key" indicator is provided for every state. We trained DPT on Key-To-Door for $150,000$ updates in Markovian and non-Markovian setups to show the difference in performance. As it can be seen in Figure 15, in the Markovian case the model converges to the optimal return, finding both a key and a door. In the latter case, the model reaches a plateau reward of 1 which means it finds a key. As we empirically observe, without the indicator DPT reaches only a suboptimal return. We believe it happens due to DPT inability to reason whether a key was already found from the random context. Without this knowledge, it is impossible for the agent to know whether it should search for the key or for the door.

We also show logs for Key-To-Door Experiments: Markovian: wandb; non-Markovian: wandb

Table 7: Q-Learning Hyperparameters

Hyperparameter	Value
Num. Train Goals	2424
Num. Histories	5000
Num. Updates	50,000
Learning Rate	3e-4

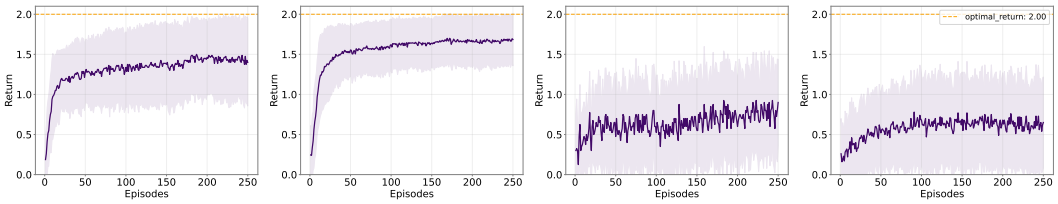

Figure 15: The performance of DPT on Key-To-Door environment. From left to right: first two plots indicate returns on 20 train and 50 test goals, respectively, for Key-To-Door as MDP by providing additional indicator of reaching the key to the model. Second two plots indicate returns with the model that has no access to the fact of reaching a key. The results are averaged across four seeds

K Additional Figures of AD Performance

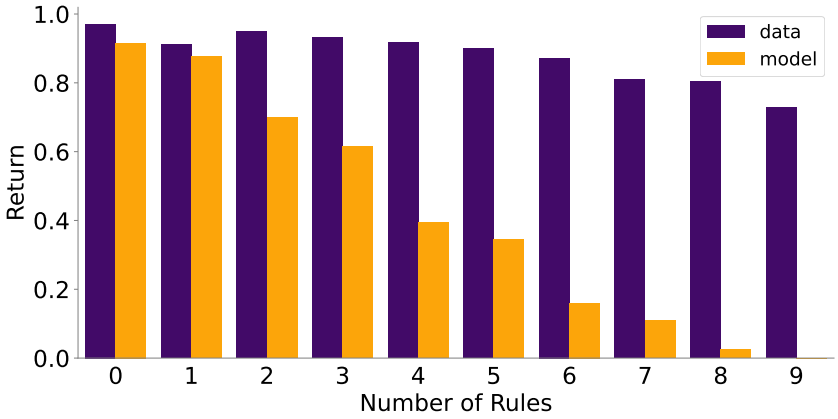

Figure 16: AD performance on different complexities of rulesets. AD is evaluated on 1024 training tasks from -100B. Sequence length is 1024.

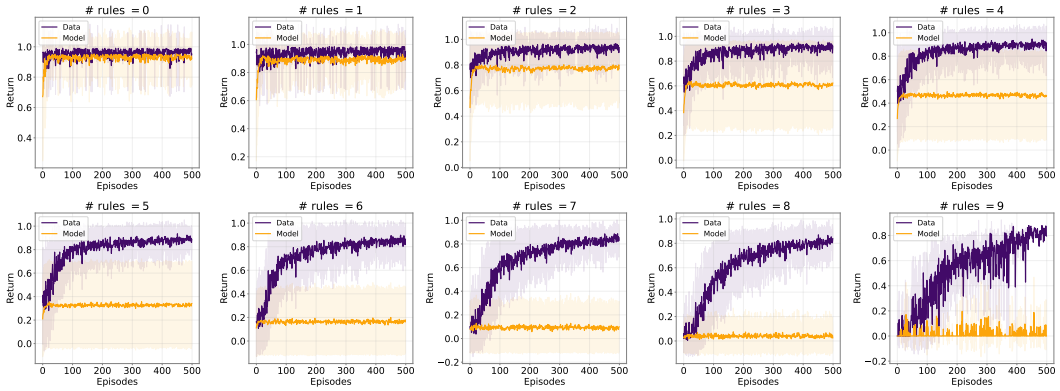

Figure 17: AD's performance on different task complexities, a full variant of Figure 6.

L Additional Figures of DPT Performance

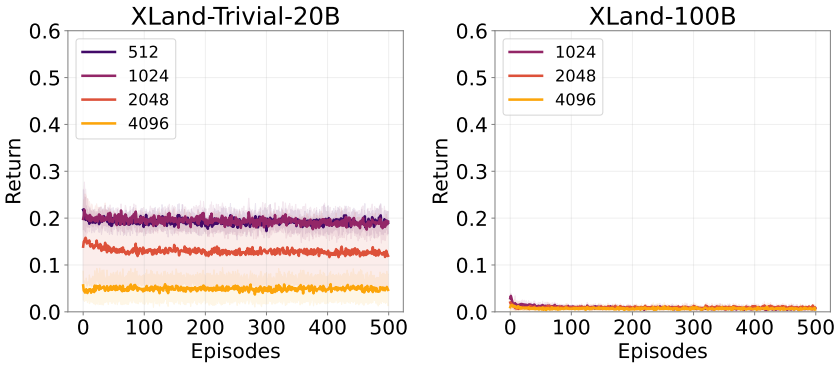

Figure 18: DPT performance on both datasets. Evaluation parameters are the same as in Figure 7.

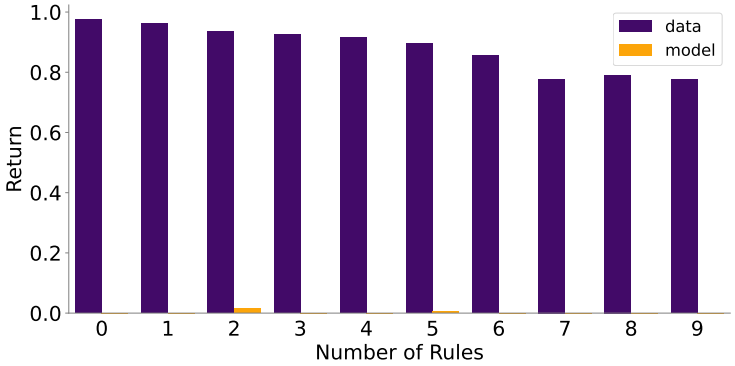

Figure 19: DPT performance on different complexities of rulesets. DPT is evaluated on 1024 training tasks from `-100B`. Sequence length is 1024.

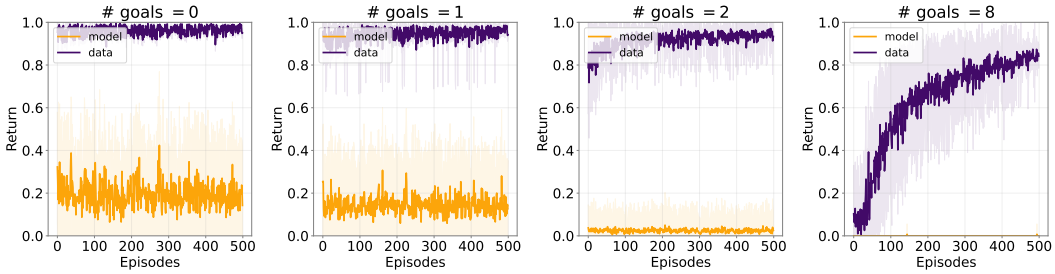

Figure 20: Comparison of the learning histories in `-100B` dataset vs. DPT performance on the same *training* tasks. DPT is not able to solve simple tasks and there is no observation in-context learning emerges, model's performance also degrades as the rulesets get harder. The context length of the model is 1024. The evaluation parameters, except training tasks, are the same as in Figure 7.

M Hyperparameters

Table 8: DPT Hyperparameters

(a) DPT Hyperparameters for datasets

Hyperparameter	Value
Embedding Dim.	64
Number of Layers	8
Number of Heads	8
Feedforward Dim.	256
Layernorm Placement	Pre Norm
Embedding Dropout Rate	0.1
Batch size	[512, 256, 128]
Sequence Length	[1024, 2048, 4096]
Optimizer	Adam
Betas	(0.9, 0.99)
Learning Rate	1e-3
Learning Rate Schedule	Cosine Decay
Warmup Ratio	0.05
# Parameters	25 M

(b) DPT Hyperparameters Key-to-Door

Hyperparameter	Value
Embedding Dim.	64
Number of Layers	4
Number of Heads	4
Feedforward Dim.	64
Layernorm Placement	Pre Norm
Residual Dropout	0.5
Sequence Length	[50, 100, 250, 350, 500]
Batch size	128
Optimizer	Adam
Betas	(0.9, 0.99)
Learning Rate	1e-3
Label Smoothing	0.3
Learning Rate Schedule	Cosine Decay
Warmup Ratio	0.05
# Parameters	200 K

Table 9: AD Hyperparameters

Hyperparameter	Value
Embedding Dim.	64
Number of Layers	8
Number of Heads	8
Feedforward Dim.	512
Layernorm Placement	Pre-norm
Embedding Dropout	0.1
Batch size	[256, 128, 64]
Sequence Length	[1024, 2048, 4096]
Optimizer	Adam
Betas	(0.9, 0.99)
Learning Rate	1e-3
Learning Rate Schedule	CosineLR
Warmup Steps	500
# Parameters	25 M

Table 10: PPO hyperparameters used in multi-task pre-training from Section 2.2.

Hyperparameter	**Value**
env_id	XLand-MiniGrid-R1-13x13
benchmark_id	medium-1m
use_bf16	True
pretrain_multitask	True
context_emb_dim	16
context_hidden_dim	64
context_dropout	0.0
obs_emb_dim	16
action_emb_dim	16
rnn_hidden_dim	1024
rnn_num_layers	1
head_hidden_dim	256
num_envs	65536
num_steps	256
update_epochs	1
num_minibatches	64
total_timesteps	25,000,000,000
optimizer	Adam
decay_lr	True
lr	0.0005
clip_eps	0.2
gamma	0.995
gae_lambda	0.999
ent_coef	0.001
vf_coef	0.5
max_grad_norm	0.5
eval_episodes	256
eval_seed	42
train_seed	42

Table 11: PPO hyperparameters used in single-task fine-tuning from Section 2.2.

Hyperparameter	**Value**
env_id	XLand-MiniGrid-R1-13x13
benchmark_id	medium-1m
use_bf16	True
pretrain_multitask	False
context_emb_dim	16
context_hidden_dim	64
context_dropout	0.0
obs_emb_dim	16
action_emb_dim	16
rnn_hidden_dim	1024
rnn_num_layers	1
head_hidden_dim	256
num_envs	8192
num_steps	256
update_epochs	1
num_minibatches	8
total_timesteps	1,000,000,000
optimizer	Adam
decay_lr	True
lr	0.0005
clip_eps	0.2
gamma	0.995
gae_lambda	0.999
ent_coef	0.001
vf_coef	0.5
max_grad_norm	0.5
eval_episodes	256
eval_seed	42
train_seed	42