N-Gram Induction Heads for In-Context RL: Improving Stability and Reducing Data Needs

Anonymous Authors[1]

Abstract

In-context learning allows models like transformers to adapt to new tasks from a few examples without updating their weights, a desirable trait for reinforcement learning (RL). However, existing in-context RL methods, such as Algorithm Distillation (AD), demand large, carefully curated datasets and can be unstable and costly to train due to the transient nature of in-context learning abilities. In this work, we integrated the n-gram induction heads into transformers for in-context RL. By incorporating these n-gram attention patterns, we considerably reduced the amount of data required for generalization and eased the training process by making models less sensitive to hyperparameters. Our approach matches, and in some cases surpasses, the performance of AD in both grid-world and pixel-based environments, suggesting that n-gram induction heads could improve the efficiency of in-context RL.

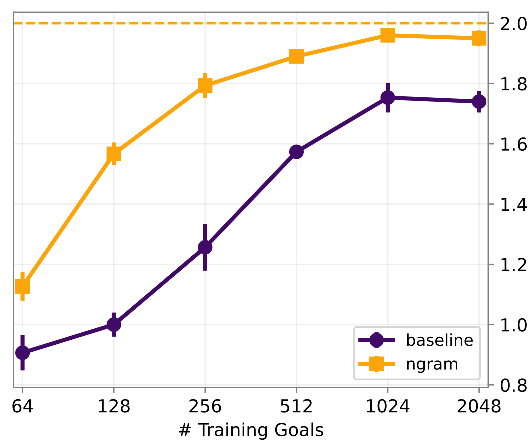

Figure 1. Performance comparison for different number of training goals between our method and Algorithm Distillation (AD), an in-context reinforcement learning method (Laskin et al., 2022). Our method demonstrates similar performance with less training goals (128 vs. 512) and in general outperforms the baseline. See Section 4 for results.

1. Introduction

In-context learning is a powerful ability of pretrained autoregressive models, such as transformers (Vaswani et al., 2023) or state-space models (Gu et al., 2022). In contrast to fine-tuning, in-context learning is able to effectively solve downstream tasks on inference without explicitly updating the weight of a model, making it a versatile tool for solving wide range of tasks (Agarwal et al., 2024). Originated in the language domain (Brown et al., 2020), the in-context ability has quickly found its applications in Reinforcement Learning (RL) for building agents that can adaptively react to the changes in the dynamics of the environment. This trait allows researchers to use In-Context Reinforcement Learning (ICRL) as a backbone for the embodied agents (Elawady et al., 2024) or to benefit from its adaptation abilities for domain recognition in order to build generalist agents (Grigsby et al., 2024).

Notably, in-context reinforcement learning methods require specifically curated data, which can be demanding to obtain (Nikulin et al., 2024b). In addition, the in-context ability itself is transient (Singh et al., 2024) and it is difficult to predict its emergence from cross-entropy loss alone (Agarwal et al., 2024), making the training of such models unstable and expensive in terms of training budget. Our work takes initial steps toward addressing these challenges by introducing modifications to the transformer's attention heads, which can accelerate training and reduce the amount of data required for in-context learning to emerge.

In our work, we propose integrating an n-gram induction head into the ICRL model. As we demonstrate, these heads can improve model performance in low-data settings and reduce hyperparameter sensitivity while introducing only a few additional hyperparameters that are straightforward to optimize. We provide experimental evidence on Dark Room, Key-to-Door and Miniworld environments, covering

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Workshop on Test-Time Adaptation. Do not distribute.

both discrete and visual observation spaces.

To summarize our main contributions, in this paper we show that **N-Gram attention heads**:

- **Decrease the amount of data needed for generalization on novel tasks.** By utilizing n-gram heads, it is possible to reduce the total number of transitions in training data by a maximum of **27x** compared to the original method of Laskin et al. (2022). The results are presented in Section 4.
- **Help mitigate hyperparameter sensitivity in ICRL models, contributing to more stable training.** By employing n-gram heads, one may need less time searching for a good set of hyperparameters. The results are presented in Section 4.
- **Can be used in the environments with visual observations.** However n-grams are originally found in discrete structures (e.g. natural language texts), we show it is possible to detect repeating patterns in the sequences of images. The details of the implementation are presented in Appendix C and the results of the experiment are shown in Section 4.

2. Method

2.1. Algorithm Distillation

We build our method on Algorithm Distillation (Laskin et al., 2022) and use it as our baseline. It is an in-context reinforcement learning algorithm that distills the policy improvement operator by training a transformer model on specifically acquired data. As training data, the authors propose to use the learning histories of many RL algorithms that are trained to solve different tasks in the multi-task environment. After pretraining on such data, the model is able to solve unseen tasks entirely in-context by interacting with an environment without explicitly updating weights of the model.

More formally, if we assume that a dataset $\mathcal{D}$ consists of *learning histories*, then

$$\mathcal{D} := \left\{ (\tau_1^g, ..., \tau_n^g) \sim \left[\mathcal{A}_g^{source} | g \in \mathcal{G} \right] \right\},$$

where $\tau_i^g = (o_1, a_1, r_1, ..., o_T, a_T, r_T)$ is a trajectory generated by a source algorithm from $\mathcal{A}_g^{source}$ for a goal g from a set of all possible goals $\mathcal{G}$, and o_i, a_i, r_i are observations, actions and rewards, respectively.

Such data might be difficult to obtain, since the aforementioned process requires training thousands of RL algorithms solving different tasks to obtain enough learning histories. In addition, AD suffers the same problems as any in-context algorithm. Learning the optimal solution can be delayed by a tendency of transformers to learn simple structures at first (Edelman et al., 2024). Moreover, the nature of in-context ability is unstable and can fade into in-weights regime as the training progresses, considerably complicating the emergence of adaptation ability (Singh et al., 2024).

2.2. N-Gram Attention

To address simplicity bias and improve data efficiency, we include an n-gram attention layer (Akyürek et al., 2024) as one of the transformer layers. This type of layer has been shown to effectively reduce simplicity bias and enhance in-context performance. Essentially, it directly incorporates the computation of n-gram statistics into the transformer, instead of relying on them to develop naturally over time. The attention pattern that is calculated from the input sequence and used in N-Gram Head (NGH) is defined as:

$$A(n)_{ij} \propto \mathbb{1}[(\wedge_{k=1}^{n} x_{i-k} = x_{j-k-1})] \, .$$

After that, we apply a projection and add a residual to the output:

$$\texttt{NGH}^n \left(h^l\right) = W_1 h^l + W_2 A(n)^\top h^l \, ,$$

where n is the length of n-grams, W_1 and W_2 are learnable projection matrices and h^l is an embedding from a previous transformer layer. In simple terms, we look for n-gram occurrences and with the help of $A(n)$ attention pattern force gradients to flow only through tokens that co-occur in the sequence.

Following Akyürek et al. (2024), we also implement an N-Gram layer, which closely resembles a traditional transformer layer. The layer consists of a head NGH^i that is processed through a MLP and then added to the residual stream:

$$\texttt{NGL}^n \left(h^l\right) = h^l + \texttt{MLP}[\texttt{NGH}^n(h^l)].$$

In the original paper, the authors used text tokens from the input sequence for n-gram matching. We lack such an opportunity when dealing with image observations, so we ought to use quantization in order to enable n-gram matching. The implementation details of the quantization process and how matching is performed are described in the Section 2.3.

2.3. N-Gram Matching

To find n-grams in environments with a *discrete observation* space, we use raw input sequence. However, since we are working in RL setting, the input sequence has a form of

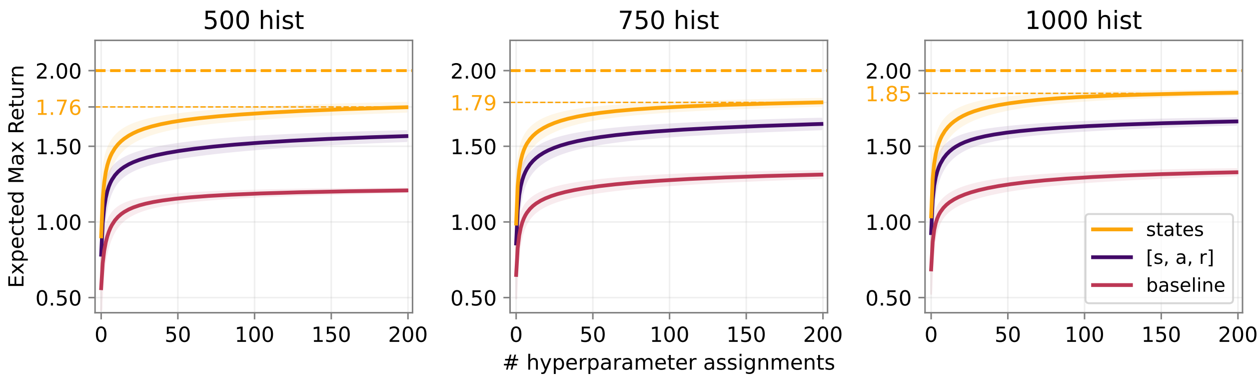

Figure 2. Results on Key-to-Door. We demonstrate the ability of our method to generalize when the task diversity is limited. We fix the total number of goals with 100, significantly shrinking the number of learning histories. Keep in mind that for the baseline method to converge to a model with the same performance, it needs 2048 goals and 2048 learning histories (Laskin et al., 2022). We show that our method needs **27x** less data comparing to baseline (see Appendix F for justification). The baseline method can no longer converge with that few data and its performance plateaus with the increasing number of hyperparameter assignments, while N-Gram model shows near-optimal performance.

$(s_0, a_0, r_0, \dots, s_n, a_n, r_0)$, so in our experiments we tested two approaches. We either compare the equivalence of full transitions $(a_{i-1}, r_{i-1}, s_i) = (a_{j-1}, r_{j-1}, s_j)$ or just states $(s_i = s_j)$.

In case of *pixel-based observations*, We cannot directly match raw images, as even slight variations can result in a mismatch. To address this, we use Vector Quantization (VQ) (van den Oord et al., 2017; Gersho & Gray, 1991) to quantize observations into the vectors from a codebook. We pretrain a ResNet (He et al., 2016) encoder-decoder model with a VQ bottleneck, which is trained to reconstruct the input image. After pretraining, each image is mapped into a 4×4 matrix of indices, and we use these for the n-gram matching. We count a match only if all the indices in the matrix are equal.

Before training starts, we use the VQ model to label images from a dataset with their indices and then train both causal and n-gram attention layers simultaneously. During the evaluation, we only make a forward pass of the VQ model in order to get the latent vectors and indices for n-gram matching.

3. Experiment Setup

3.1. Evaluation Protocol

We set up and follow a specific evaluation protocol to showcase the benefits of using N-Gram layers in the ICRL setting. We use a random search over the hyperparameter space. Reporting aggregated hyperparameter search results instead of cherry-picking the best runs allows us to demonstrate the hyperparameter sensitivity of each method. To ensure that in each experiment a model has processed an equal amount of data, we fixed the batch size and limited the number of gradient steps during a run to 10K.

We evaluate the models on previously unseen goals that were not included in the training dataset. In the Dark Room environment, the number of evaluation goals varies across experiments and corresponds to all goals excluded from the training set. For instance, if a model is trained on 20 goals, it is evaluated on the remaining 60 goals. For Key-to-Door evaluation, we use 100 unseen goals and 50 unseen goals for Miniworld Key-to-Door.

To show the difference between our method and the baseline, we choose to report the Expected Maximum Performance metric (EMP) (Dodge et al., 2019; Kurenkov & Kolesnikov, 2022). By doing so, we do not report the best performance of a single checkpoint, rather we show the expected maximum performance for a certain computational budget. Using this approach, we simultaneously compare our method with a baseline in terms of ease of training and maximum achieved performance. The exact hyperparameter assignment setups are shown in Appendix G.

4. Results

N-Gram layers can make the search for optimal hyperparameters quicker. To demonstrate the effect of N-Gram layers on the hyperparameter sensitivity of the model, we perfrom a random search over the core transformer hyperparameters that do not change the parameter count of the model. The effect of N-Gram heads is illustrated in Figure 6. In the top row, we fix the number of training tasks at 60 and vary the number of learning histories. It can be seen that the

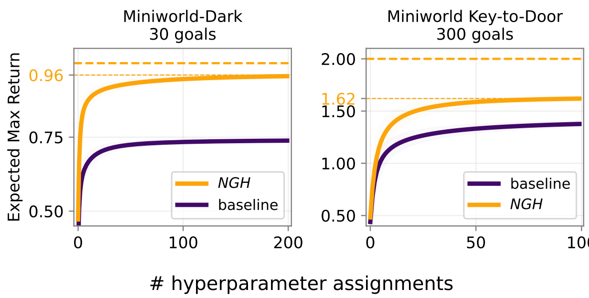

Figure 3. Results on Miniword environments. We show that our method is applicable not only for environments with discrete observations, but also for the image-based ones. The settings of the Miniworld environments are similar to Dark Room and Key-to-Door. The main outcome of these experiments is that we can successfully implement n-gram matching in for images and get similar results to the discrete environments. The details of the setup are described in Section 4.

model with n-gram layers can find the optimal parameters faster than the baseline model. For 1K learning histories, finding the optimal model requires just over 20 hyperparameter assignments, while the baseline model needs more than 400. When the number of tasks varies, the baseline model quickly saturates at suboptimal performance and asymptotically improves thereafter, whereas the n-gram model reaches optimal performance in about 15 assignments. Full-length plots are available in Figure 9.

N-Gram layers improve data-efficiency of ICRL algorithm. In real-world data, there are often many trajectories per task, but the number of distinct tasks is limited. (Yu et al., 2019; Gallouédec et al., 2024). In such cases, a desirable quality of the model is its ability to avoid overfitting on the training data while generalizing to unseen tasks. Our hypothesis here is that incorporating N-Gram layers into the model can help build a more data-efficient model and enhance generalization by capturing sequential patterns within trajectories.

To show the effect of N-Gram layers when task diversity in data is low, we set up an experiment in the Key-to-Door environment, since it possesses 6.5K tasks in total. To simulate low task diversity, we fix the number of training goals by 100 and sample another 100 unseen goals for evaluation. It can be observed from Figure 2 that the baseline method is struggling to produce a model that is able to generalize to unseen goals in such a low data setting. In turn, our method demonstrates performance on par with what Laskin et al. (2022) report in their work. We note that compared to AD, our method needs 27x less data, detailed computations are provided in Appendix F.

N-Gram layers can be used with images as observations. It is relatively straightforward to match n-grams in discrete settings, like text or grid-world environments. The problem arises when the observation space is image-based. We cannot directly compare the images, as even a slight camera rotation would invalidate a match; however, they may still correspond to the same state.

To address this, we need a model that disregards minor differences in its encoding and instead focuses on state-representative details, such as the color of the wall the agent sees and its distance from the wall. We utilize the Vector Quantization (van den Oord et al., 2017) technique for this reason, the details of n-gram matching are described in Section 2.3.

We transfer the Dark Room and Key-to-Door setting into a 3D environment Minigrid, where an agent receives a $3 \times 64 \times 64$ RGB image as an observation. We observed similar differences in performance of the N-Gram and baseline models. N-Gram layer is able to reduce the number of hyperparameter assignments needed to find a model with near-optimal performance in both Miniworld-Dark (Room, omitted for brevity) and Miniworld-Key-to-Door environments, see Figure 5. In a low-data regime, N-Gram layers also improve performance compared to the baseline. As shown in Figure 3, N-Gram layers enhance performance in both environments.

5. Conclusion and Future Work

In our work we show that incorporating n-gram induction heads can sufficiently ease training of in-context reinforcement learning algorithms. Our findings are threefold: (**i**) we show that n-gram heads can fairly decrease sensitivity to hyperparameters of ICRL models; (**ii**) we demonstrate that our method is able to generalize from much fewer data than the baseline Algorithm Distillation (Laskin et al., 2022) approach. (**iii**) however the original n-gram heads were designed for discrete spaces, we showed it is possible to adapt the approach to environments with visual observations by utilizing vector quantization techniques. We speculate that n-gram heads are useful in ICRL due to the imperfect nature of in-context learning itself: a tendency of transformers to converge to simple solutions first (Edelman et al., 2024), and the transitivity of the in-context ability itself (Singh et al., 2024).

Although we believe our findings are promising, there are some limitations of the current work. Further research is needed to investigate the behavior of N-Gram heads in more comprehensive environments, e.g. XLand-Minigrid (Nikulin et al., 2024a) or Meta-World (Yu et al., 2019). Additionally, while image observations account for a significant portion of RL applications, exploring methods to apply N-Gram heads to proprioceptive continuous states could provide further insights.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, R., Singh, A., Zhang, L. M., Bohnet, B., Chan, S., Anand, A., Abbas, Z., Nova, A., Co-Reyes, J. D., Chu, E., et al. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*, 2024.

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

Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., and Mercer, R. L. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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, 2021. URL https://arxiv.org/abs/2106.01345.

Dodge, J., Gururangan, S., Card, D., Schwartz, R., and Smith, N. A. Show your work: Improved reporting of experimental results. *arXiv preprint arXiv:1909.03004*, 2019.

Edelman, B. L., Edelman, E., Goel, S., Malach, E., and Tsilivis, N. The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*, 2024.

Elawady, A., Chhablani, G., Ramrakhya, R., Yadav, K., Batra, D., Kira, Z., and Szot, A. Relic: A recipe for 64k steps of in-context reinforcement learning for embodied ai, 2024. URL https://arxiv.org/abs/2410.02751.

Gallouédec, Q., Beeching, E., Romac, C., and Dellandréa, E. Jack of all trades, master of some, a multi-purpose transformer agent, 2024. URL https://arxiv.org/abs/2402.09844.

Gersho, A. and Gray, R. M. Vector quantization and signal compression. In *The Kluwer International Series in Engineering and Computer Science*, 1991. URL https://api.semanticscholar.org/CorpusID:118950728.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Grigsby, J., Sasek, J., Parajuli, S., Adebi, D., Zhang, A., and Zhu, Y. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. *arXiv preprint arXiv:2411.11188*, 2024.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kirsch, L., Harrison, J., Freeman, D., Sohl-Dickstein, J., and Schmidhuber, J. Towards general-purpose in-context learning agents. Workshop on Distribution Shifts, 37th Conference on Neural Information . . . , 2023.

Kneser, R. and Ney, H. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pp. 181–184 vol.1, 1995. doi: 10.1109/ICASSP.1995.479394.

Kurenkov, V. and Kolesnikov, S. Showing your offline reinforcement learning work: Online evaluation budget matters. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11729–11752. PMLR, 17–23 Jul 2022.

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.

Liu, J., Min, S., Zettlemoyer, L., Choi, Y., and Hajishirzi, H. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint arXiv:2401.17377*, 2024.

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

Nikulin, A., Kurenkov, V., Zisman, I., Agarkov, A. S., Sinii, V., and Kolesnikov, S. Xland-minigrid: Scalable meta-reinforcement learning environments in jax. In *Automated Reinforcement Learning: Exploring Meta-Learning, AutoML, and LLMs*, 2024a.

Nikulin, A., Zisman, I., Zemtsov, A., Sinii, V., Kurenkov, V., and Kolesnikov, S. Xland-100b: A large-scale multi-task dataset for in-context reinforcement learning. *arXiv preprint arXiv:2406.08973*, 2024b.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Roy, A., Anil, R., Lai, G., Lee, B., Zhao, J., Zhang, S., Wang, S., Zhang, Y., Wu, S., Swavely, R., et al. N-grammer: Augmenting transformers with latent n-grams. *arXiv preprint arXiv:2207.06366*, 2022.

Schmied, T., Paischer, F., Patil, V., Hofmarcher, M., Pascanu, R., and Hochreiter, S. Retrieval-augmented decision transformer: External memory for in-context rl, 2024. URL https://arxiv.org/abs/2410.07071.

Singh, A., Chan, S., Moskovitz, T., Grant, E., Saxe, A., and Hill, F. The transient nature of emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. Corl: Research-oriented deep offline reinforcement learning library. *Advances in Neural Information Processing Systems*, 36, 2024.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

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 (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

Zisman, I., Kurenkov, V., Nikulin, A., Sinii, V., and Kolesnikov, S. Emergence of in-context reinforcement learning from noise distillation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

A. Related Work

In-context RL. The key feature behind ICRL is the adaptation ability of a pretrained agent. In general, it relies on the transformer's ability to infer a task from the history of interactions with an environment. Müller et al. (2021) show that transformers are capable of Bayesian inference, which is known for its applicability to reasoning under uncertainty (Ghavamzadeh et al., 2015). Laskin et al. (2022) proposed to pretrain a transformer on the learning histories of RL algorithms which allows it to implicitly learn the policy improvement operator. During inference on unseen tasks, a transformer is able to improve its policy by observing a context and inferring a task from it. However, such an approach requires specific datasets, which may be expensive to collect (Nikulin et al., 2024b). To address this, it has been proposed to generate datasets following the noise curriculum instead of training thousands of RL agents (Zisman et al., 2024), perform augmentations of existing data (Kirsch et al., 2023) or filter out irrelevant data (Schmied et al., 2024). Our work follows the direction of democratizing data restrictions, but instead of working with data, we introduce a model-centric approach, making a transformer to perform in-context reinforcement learning with less data.

N-Gram and Transformers. N-Gram statistical models have been known for decades and used in the statistical approach to language modeling (Brown et al., 1992; Kneser & Ney, 1995). More recent approaches (Roy et al., 2022; Liu et al., 2024) study the application of n-grams to transformer models, finding that they can increase overall performance. Akyürek et al. (2024) discover that a transformer implicitly implements the 2-gram attention pattern when solving the in-context learning task, which authors denote as a higher order of induction head (Olsson et al., 2022). They explicitly implement 1-, 2-, and 3-gram attention layers and observe a significant reduction in perplexity of the pretrained models. Another work (Edelman et al., 2024) directly investigates the behavior of n-gram induction heads during the training process. The authors find that transformers are biased towards simple solutions, thus making it problematic for higher-order induction heads to appear. To our knowledge, we are the first to apply these findings in a decision-making setting.

B. Environments

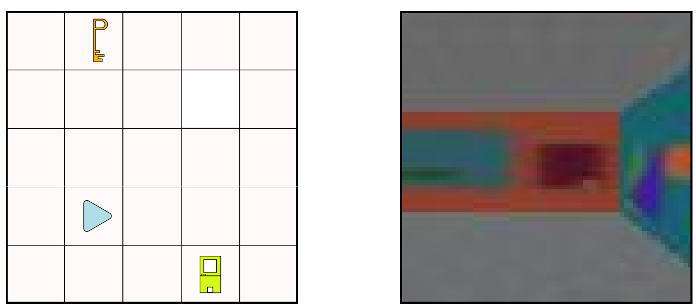

Figure 4. **(Left)** The Key-to-Door environment. The key and the door are shown for illustrative purposes only; the agent does not see their location during training. **(Right)** An observation from the Miniworld environment.

Dark Room is an MDP grid-world environment with discrete state and action spaces The grid size is 9×9, where an agent has 5 possible actions: up, down, left, right and do nothing. The goal is to find a target cell, the location of which is not known to the agent in advance. The episode length is fixed at 50 time steps, after which the agent is reset to the middle of the grid. The reward $r = 1$ is given for every time step the agent is on the goal grid, otherwise $r = 0$. The agent does not know the position of the goal, hence it is driven to explore the grid. The environment consists of 80 goals in total, excluding the starting square.

Dark Key-to-Door is a POMDP environment, similar to Dark Room, but with a more complicated task. The agent first needs to find a square with a key, and then only to find a door. The reward is given when the key is found ($r = 1$) and once the door is opened (also $r = 1$), after which the episode ends. The agent then resets to a random grid. The maximum episode length is 50, and since we can control the location of the key and door, there are around 6.5k possible tasks. The key difference of Key-to-Door compared to Dark Room is that an agent needs to use the memory to recall whether or not the key was collected to adapt its exploration strategy and successfully solve the task. We do not provide any hints after the key was collected, which makes the environment only partially observed.

Both *Key-to-Door* and *Dark Room* serve as a good starting point for testing the in-context ability in an RL setting. Despite its simple grid-structure, AD still needs a substantial amount of data to start showing decent performance, and these environments serve as a testbed to show N-Gram Layers help with data efficiency.

Miniworld is a 3D environment with an RGB 64×64 images as observations and a discrete action space. We test our method in two settings of Miniworld, the first resembling Dark Room and the latter Key-to-Door. The agent can perform three actions: move a step ahead and turn the camera left or right, no lateral movement is allowed. The episode length is 50 for Miniworld-Dark Room and 100 for Miniworld-Key-to-Door.

The Miniworld-based environments are of special interest, because while it was trivial to search for n-grams with discrete states, pixel-based observations are not so easily comparable. The details of n-grams matching for Miniworld are described in Section 2.3

C. Implementation Details

We take a GPT-2 as a backbone, borrowing the implementation of a Decision Transformer (Chen et al., 2021) from the CORL library (Tarasov et al., 2024). As input, the transformer receives a tuple (a_{i-1}, r_{i-1}, s_i) of actions, rewards, and observations that are combined into a single token through a linear map to reduce the length of the sequence. The implementation of the N-Gram layer is taken from (Akyürek et al., 2024) with a few minor modifications to match our implementation of a causal transformer layer. The parameter count of our model is 20M.

For *Dark Room* and *Key-to-Door* we use a simple embedding layer to map states, actions, and rewards into the model space. For *Miniworld*-based environments, we pretrain a VQ model on images (as described in the previous subsection) and use its encoder to embed pixel-based observations into latent vectors. Actions and rewards are processed using an embedding layer. We set the context length of the transformer so that there are at least two episodes in it, to maintain cross-episodic context.

To ensure that our implementation of a baseline (AD) can solve the environments, we present the performance of a baseline that is trained on optimal hyperparameters in Appendix I.

D. Data Collection

Algorithm Distillation introduce several requirements on the structure of the data. It should be comprised of learning histories, i.e. there should be an implicit ordering in data from the least to the most effective policy. To produce such histories, we used a combination of approaches.

For grid-world environments, we use a table Q-Learning algorithm (Watkins & Dayan, 1992) and save (s_i, a_i, r_i) transitions. In image-based environments, we use the approach described in Zisman et al. (2024). For this, we implement an oracle agent and design a decaying noise schedule. It allows us to collect the learning histories faster than training any model-free RL algorithm from scratch for each task. The rest of the data collection process remains unchanged.

Throughout the text we use the terms *learning histories* and *tasks*. The task is a predefined grid or a pair of grids an agent must come to upon it receives a reward. The learning history is an ordered collection of states, actions and rewards an RL algorithm observed (or produced) while learning to solve a *single* task. When we say we generated a dataset of n tasks with m learning histories, it means for each of the task there are at least $\lfloor \frac{m}{n} \rfloor$ learning histories per task. Unlike Laskin et al. (2022), we distinguish between tasks and learning histories, as it is often the case with real data when many trajectories correspond to only a few tasks (Yu et al., 2019; Gallouédec et al., 2024).

E. Additional Experiments

E.1. N-Gram layers do not significantly expand hyperparameter search space

N-Gram layer introduce new hyperparameters to optimize, such as n-gram length and position of the layer to which N-Gram layer is inserted. A natural question arises: do these hyperparameters also require extensive search, and how sensitive is the

Table 1.

(a) Ablation on n-gram length

N-Gram max	EMP
1-gram	0.74 ± 0.02
2-gram	0.71 ± 0.01
3-gram	0.76 ± 0.05

(b) Ablation on N-Gram layer position

Position	EMP
[1]	0.69 ± 0.03
[2]	0.69 ± 0.02
[1, 2]	0.67 ± 0.005

(c) Comparison of baseline and a random n-gram mask

Model	EMP
Permuted	0.51 ± 0.03
Baseline	0.52 ± 0.02

model to them?

To address this question, we conducted six random hyperparameter searches in Miniworld-Dark, ablating either the layer position or the n-gram length while keeping one variable fixed. For the n-gram length search, we fixed the position at [1] (after the first layer), whereas for the layer position HP search, we set the n-gram length to 1. Following (Akyürek et al., 2024), we do not insert N-Gram layer as the first or last layer. While searching for the optimal n-gram length, we consider ”up to” a given n-gram. For example, a 2-gram includes both a 1-gram and a 2-gram together. We continue to report the EMP metric, but here we present only the final value achieved after all hyperparameter assignments (full plots are available in Appendix H).

Table 1(a) and Table 1(b) show that there is no significant difference between neither the n-gram length, nor the position of the N-Gram layer inside a transformer. This may indicate that there is little to no overhead in hyperparameter search caused by introduction of N-Gram layers.

E.2. Inserting N-Gram layers does not hurt the performance of a baseline algorithm

Another concern when working with N-Gram layers is whether they can affect the performance of a baseline model. Hypothetically, this can occur if the quantization model fails to correctly identify which image observations correspond to the same underlying state, rendering the n-gram matching mechanism ineffective.

We designed the following experiment to test this hypothesis. Using VQ as an n-gram extraction tool, we follow the standard procedure described in Section 2.3, with one key modification. After matching, we shuffle the n-gram attention matrix $A(n)_{ij}$, effectively simulating a completely ineffective N-Gram attention layer that selects incorrect observations as n-gram matches. Like in the previous experiment, we run a random HP search in Miniworld-Dark environment and report the EMP calculated for the last hyperparameter assigned.

We compare the model with the permuted n-gram mask with the baseline model without the N-Gram layer, the results are shown in Table 1(c). No significant difference is observed between the two models, suggesting that when the n-gram matching mechanism is flawed, the model's performance remains comparable to that of a model without an N-Gram layer.

F. Calculation of Transitions in Data

In appendix I of Laskin et al. (2022) they mention that AD is more data-effective than source algorithm and report the size of a dataset. The total number of data needed to achieve an approximate of 1.81 return on Key-to-Door [1] is reported as

> (...) on 2048 Dark Key-to-Door tasks for 2000 episodes each.

The estimate of total number of transitions *to generate* for AD, considering the maximum length of an episode in Key-to-Door is 50 steps, equals: $2048 \times 2000 \times 50 = 204.8\texttt{M}$ transitions.

We generate 100 unique training tasks and then sample 750 train task with repetition from the original 100. Then we make 200 training episodes for each task. In total, we get $750 \times 200 \times 50 = 7.5\texttt{M}$ transitions, which is more than **27x** less data.

[1] since no accurate data of plots was published, we used free-to-use WebPlotDigitizer for Fig. 6 in AD paper

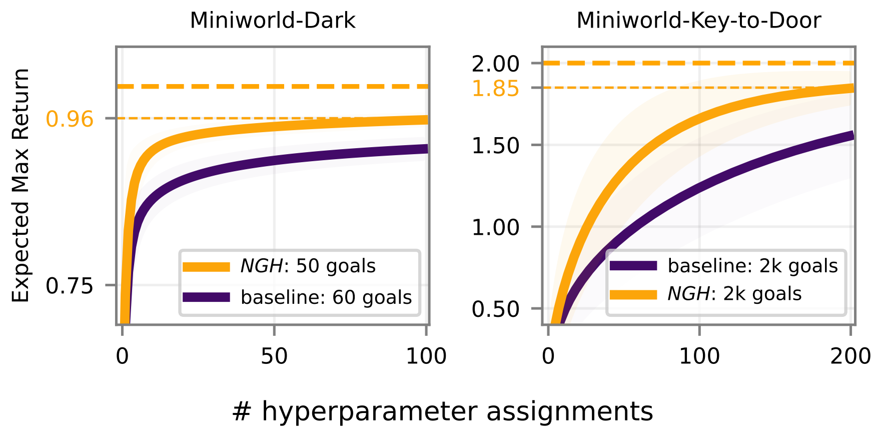

Figure 5. Experiments on hyperparameter sensitivity in 3D environments. We show the same pattern that was observed in discrete environments: the N-Gram layer makes the hyperparameter search faster than for the baseline model. More details about environments and n-gram matching are described in Section 4. **(Left)** Results on Miniworld-Dark. The N-Gram layer model is trained on 50 goals, the baseline model is on 60. For evaluation, 20 goals were used. **(Right)** Results on Miniworld-Key-to-Door. Both N-Gram and baseline models were trained on 2K goals and evaluated on 100 unseen goals.

G. HP Search Setups

We use weights and biases sweep for running sweeps. All of the sweep setups are available by this clickable link [will be available for camera-ready version].

We also report the setup of hyperparameter sweep in the table below.

Table 2. Hyperparameter Configurations

(a) Grid Environments

Parameter	Distribution	Values
batch size	-	1024
embedding dropout	Uniform	[0.0, 0.9]
seq len	-	[60, 100, 160, 200]
subsample	-	[4, 8, 10, 20, 50]
residual dropout	Uniform	[0.0, 0.5]
ngram head pos	-	[1], [2], [1, 2]
ngram max	-	[1, 2]
label smoothing	Uniform	[0.0, 0.8]
learning rate	Log Uniform	[1e-4, 1e-2]
weight decay	Log Uniform	[1e-7, 2e-2]
pre norm	-	[true, false]
normalize qk	-	[true, false]
hidden dim	-	512
update steps	-	10000

(b) MiniWorld environments

Parameter	Distribution	Values
batch size	-	1024
embedding dropout	Uniform	[0.0, 0.8]
seq len	-	[100, 150, 200]
subsample	-	[8, 16, 32]
residual dropout	Uniform	[0.0, 0.8]
ngram head pos	-	[1], [2], [1, 2]
ngram max	-	[1, 2]
label smoothing	Uniform	[0.0, 0.8]
learning rate	Log Uniform	[5e-4, 1e-2]
weight decay	Log Uniform	[1e-7, 2e-2]
pre norm	-	[true, false]
normalize qk	-	[true, false]
hidden dim	-	512
update steps	-	10000

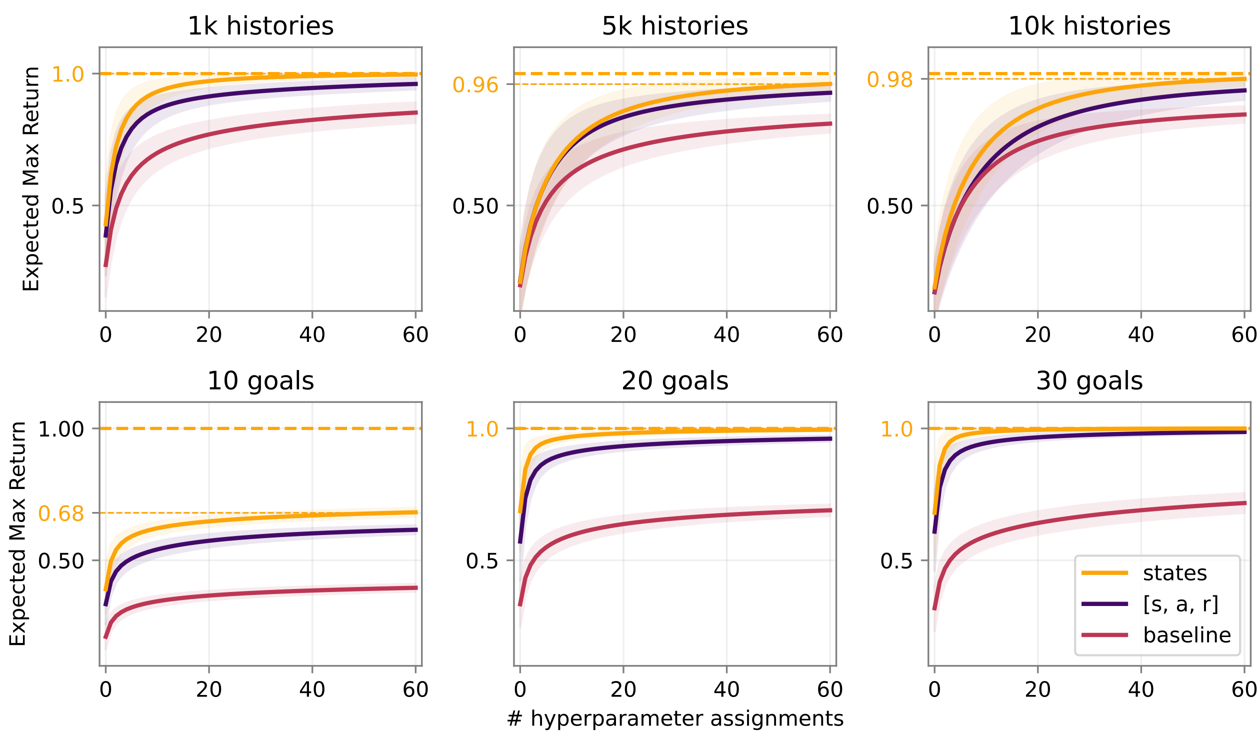

Figure 6. Results on Dark Room. We search through hyperparameters in random order and report expected maximum performance (Dodge et al., 2019). We also constrain the number of optimization steps by 10K and use equal batch size to ensure both methods use the same amount of data. **The top row** shows experiments with different number of learning histories, with the total number of training goals fixed. It is seen that our method needs much less hyperparameter assignments (20 for 1K histories) to find the optimal model, while the baseline performance increases only asymptotically (full plots are shown in Appendix H). The number of traning tasks for this experiment is 60. **The bottom row** presents experiments with varied number of goals and fixed number of learning histories. Our method makes it possible to find the optimal hyperparameters with only 15 hyperparameter assignments, while the baseline fails to work in such low data conditions. However, none of the methods can learn to generalize from only 10 goals. The number of learning histories for this task is 1K.

H. Full Plots

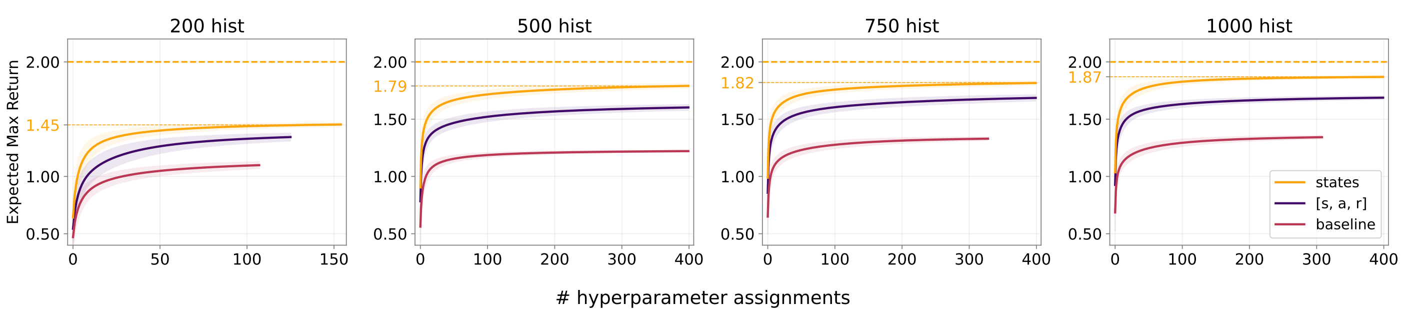

Figure 7. Full length plots for Key-to-Door. For 200 learning histories we halted the random search early, since it was obvious the performance has stalled.

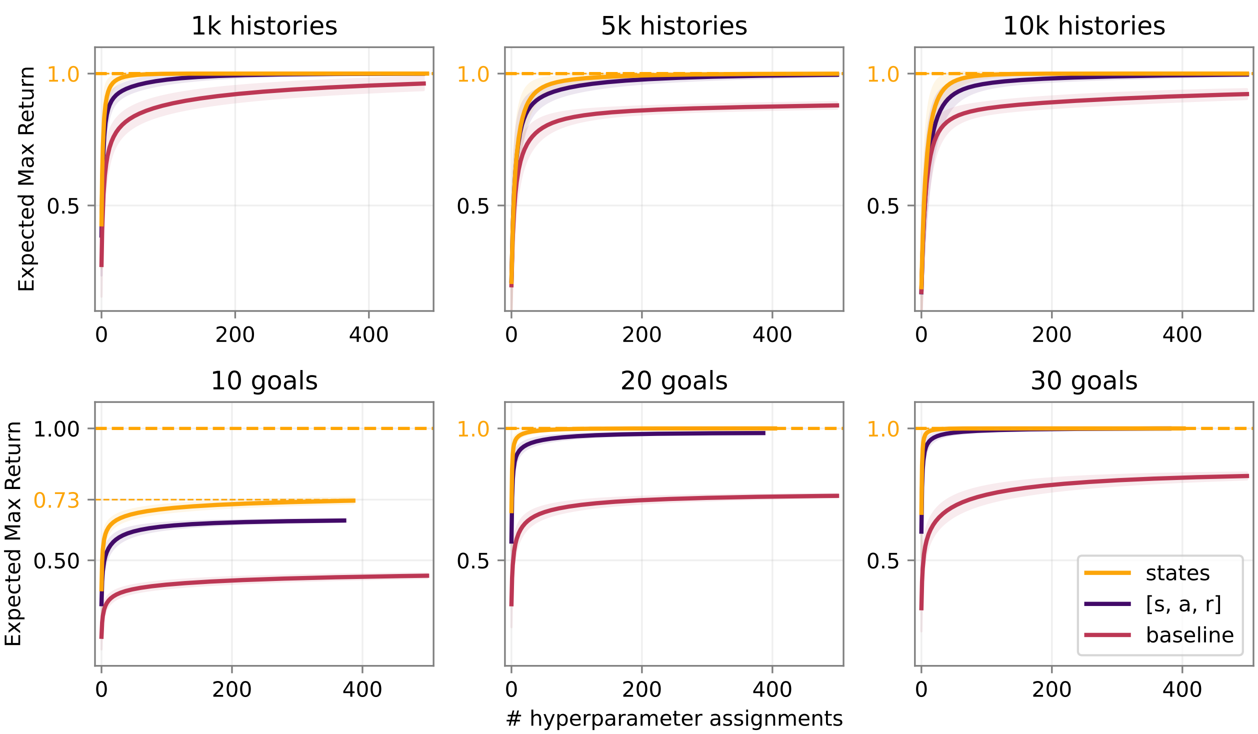

Figure 8. Full length plots for Dark Room. Some of the computations halted earlier for the same reason as in Figure 7

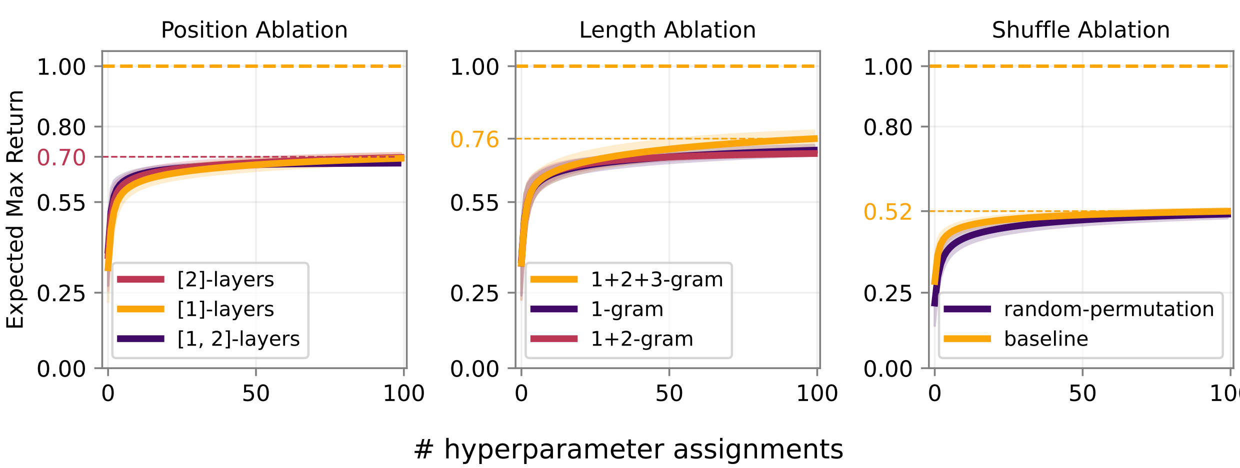

Figure 9. Full length plots for ablation experiments in Miniworld-Dark environment.

I. Performance of AD on Key-to-Door and Dark Room

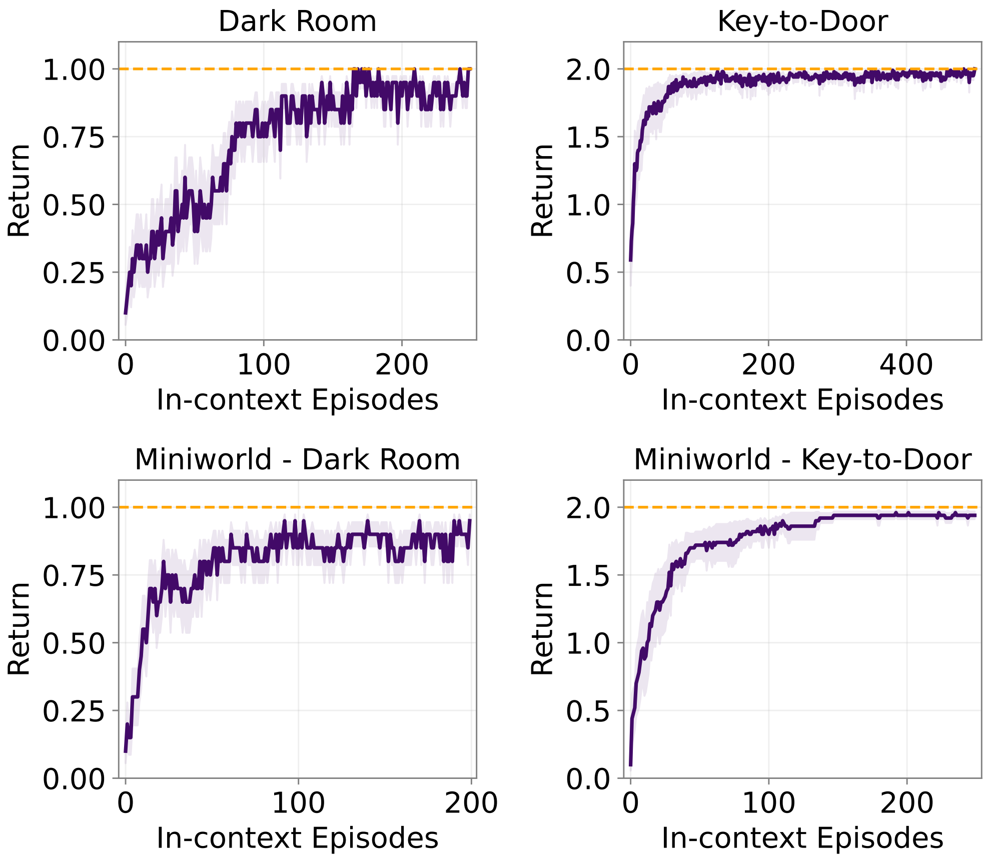

Figure 10. AD performance on Dark Room and Key-to-Door. This plot shows that our implementation of AD demonstrates optimal performance given the right hyperparameters.