Position: Reinforcement Learning Foundation Models Should Already Be A Thing

Abdelrahman Zighem [1 2] **Jill-Jênn Vie** [2]

Abstract

Foundation models for language and vision are powered by internet-scale data, while structured domains such as tabular prediction are powered by synthetic data. This substitute shifts the challenge from collection to prior design. Such priors already exist for many structured tasks: TabPFN and its successors solve tabular classification with a transformer pretrained on a synthetic Bayesian prior.

We make two points. **First**, reinforcement learning is the conspicuous gap: sampling a synthetic MDP is as feasible as sampling a synthetic tabular dataset, yet no in-context RL work treats prior design as a primary objective. **Second**, MDPs admit a fixed-size sufficient statistic, independent of the episodes observed and tabular in shape, which makes them directly amenable to the attention-based architectures used for tabular foundation models, with a policy head replacing the supervised target. Together these define the agenda for an RL foundation model.

As a proof of concept, we train a Graph Attention Network entirely on synthetic MDPs and show that, with no task-specific tuning, it solves held-out tabular benchmarks in context, both online and offline: online, in far fewer episodes than UCB-VI and tabular Q-learning, and offline, competitively with VI-LCB.

1. Foundation Models Need Strong Priors

The foundation-model paradigm (pretrain on a vast distribution, adapt to specific instances) has spread well beyond language (Bommasani et al., 2021). TabPFN, TabICL and their successors (Hollmann et al., 2023; 2025; Qu et al., 2025; 2026) showed that tabular classification problems can be solved by transformers trained on a synthetic prior over Bayesian classifiers. Prior-fitted network (PFN) training (Müller et al., 2022) has since been argued to be the dominant paradigm for Bayesian inference in any domain where a tractable prior can be sampled (Müller et al., 2025).

The recipe has three requirements: a data distribution that is cheap to sample, expressive enough to cover the deployment regime, and sufficiently structured for learning useful inductive biases. No such prior exists for reinforcement learning, so building one is an important next step for the field.

2. The Missing Prior for Reinforcement Learning

There is already solid theoretical support for the idea that in-context reinforcement learning is a viable and powerful method, even if its practical applications are still being explored or developed. For instance, Lin et al. (2024) prove that transformers with ReLU attention can efficiently implement near-optimal online RL algorithms (UCB-VI, Thompson sampling, LinUCB) purely in context, via supervised pretraining on offline trajectories. What the field has not provided is a prior that makes pretraining useful across arbitrary MDPs.

The question of what prior to place over MDPs is not new. Strens (2000) and Dearden et al. (1998) put it at the center of Bayesian RL, proposing Gaussian reward priors and Dirichlet transition priors as tractable choices. That thread has been almost entirely dropped by the modern in-context RL and meta-RL literature (Wang et al., 2016; Laskin et al., 2023; Schiff et al., 2025; Lee et al., 2023; Grigsby et al., 2024; Son et al., 2025; Lin et al., 2024). One partial exception is Duan et al. (2016), whose experimental setup does sample random MDPs from a Gaussian/Dirichlet prior, but this point is only briefly discussed in the paper.

Some recent work uses LLMs as priors (Choi et al., 2022; Yan et al., 2025), querying them to propose reward functions or action distributions. This shifts rather than solves the problem: a full LLM forward pass per sample makes large-

[1]École normale supérieure de Paris, PSL University, Paris, France [2]Soda team, Inria Saclay, Palaiseau, France. Correspondence to: Abdelrahman Zighem <abdelrahman.zighem@ens.psl.eu>.

The Workshop on Graph Foundation Models at the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea. Copyright 2026 by the author(s).

scale synthetic pretraining impractical.

All these papers use trajectories as training data, from hand-picked task families (bandit suites, procedurally generated platform games (Cobbe et al., 2020), robotic-control benchmarks), producing models adapted to a specific environment family rather than a foundation model in the sense of Bommasani et al. (2021). Sequence-modeling approaches such as the Decision Transformer (Chen et al., 2021) and Gato (Reed et al., 2022) likewise treat data as given. None of these works ask what distribution over environments would make pretraining optimal.

There is a second, independent reason to move away from raw trajectories: they do not scale. Recurrent approaches are bounded by vanishing and exploding gradients over long sequences (Bengio et al., 1994), which caps the effective memory regardless of model size. Transformer approaches avoid that failure mode but introduce another: attention cost is quadratic in context length (Vaswani et al., 2017), and the context for N episodes of horizon H has $\Theta(NH)$ tokens. As experience accumulates, managing this growing context becomes increasingly difficult, and the part of the history most at risk is the early, exploratory phase, which carries the most information about the MDP. Context compression rather than context truncation is an active research direction. Recent work on end-to-end compression for tabular foundation models (Zabërgja et al., 2026) shows that latent compression can reduce memory and inference costs by orders of magnitude, so the problem may eventually become more tractable. For now it remains a genuine obstacle, especially in the case of RL where trajectories can be numerous, long, and of various length.

3. MDPs as Graphs, and their Tabularization

A Markov Decision Process (MDP) $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ is a graph: states are nodes, actions label outgoing edges, and the Markov kernel $P(\cdot \mid s, a)$ gives weighted adjacency. Existing in-context RL approaches feed the model the *trajectory* traced over this graph, a sequence of length $\Theta(NH)$ for N episodes of average horizon H.

For small finite MDPs, the Markov property completely eliminates the need for trajectory-length context. The trajectories with a tabular MDP collapse into three sufficient statistics: visit counts $N(s, a) \in \mathbb{N}$ for each pair, empirical mean rewards $\hat{r}(s, a) \in [0, 1]$ and empirical transition row $\hat{P}(\cdot \mid s, a) \in \Delta(\mathcal{S})$, with the convention that unvisited rows default to uniform transitions and rewards are initialized at 0. Stacked across (s, a) pairs, these form a matrix

$$Z = \big(N(s,a),\ \hat{r}(s,a),\ \hat{P}(\cdot \mid s,a) \big)_{(s,a)\in\mathcal{S}\times\mathcal{A}} \qquad (1)$$

of shape $|\mathcal{S}||\mathcal{A}| \times (|\mathcal{S}| + 2)$, with total size $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ and crucially *independent of N and H*. Each row of Z is one state–action pair. The visit count and empirical reward are scalar features. The empirical transition row is an $|\mathcal{S}|$-dimensional feature vector that encodes graph adjacency and edge weights. This is the format consumed by a permutation-equivariant set-transformer of the TabPFN family.

When the state space grows, or when it is infinite, the cost of storing $\mathcal{O}(S^2A)$ entries becomes unmanageable. One can imagine various options to generalize this approach: mesh discretization and clustering are two obvious candidates. However, discretization suffers from the curse of dimensionality, and clustering may require subtle feature engineering when the data is hard to separate. We leave further exploration in this direction to future work.

Adopting the tabular *representation* does not mean adopting TabPFN as a model. PFN-style in-context learning is supervised: it predicts a target column of the input given the others. The output of the RL model is a policy over actions. The tabular framing provides the input format, the synthetic-prior training recipe, and some architectural ideas, but not the loss or the output heads. We turn to those next.

4. A Foundation Model for small tabular MDPs

In this section, we describe an architecture for solving small RL problems that builds upon the ideas described in this paper. The implementation, training code, and evaluation scripts are available at `https://github.com/Shika-B/One-Shot-Reinforcement-Learning`.

4.1. Design

A Graph Attention Network (GAT, Veličković et al. (2018)) takes $[\log(1 + N), \hat{r}]$ as features and $\hat{P}$ as multiplicative attention-biases, produces a per-row representation of the features and exposes a policy head that outputs, for a given state s, an estimate $\pi_{\text{model}}(s)$ of the probability vector $\pi^\star(s)$. The representation is refined by K propagation layers, each a single round of message passing along the transition structure. These layers are weight-tied, so K is not baked into the parameters: it can be increased or decreased at evaluation time to spend more or less compute on planning depth, without retraining. Permutation equivariance over rows gives automatic invariance under relabelings of states and actions. No positional encoding is used, so the integer identity of a given state in one MDP does not leak to another. Since MDPs in a batch differ in size, states and actions are padded to fixed maxima $S_{\max}$ and $A_{\max}$, and a state/action mask keeps this padding out of every attention, update, and policy output (cf. Appendix A).

Using a GAT allows the model to be structurally aware of the underlying graph structure of the MDP, rather than learn

it. Each round of message passing propagates information about state-action pairs backward through the graph: when the model learns that a pair (s, a) is a good target, it should also raise its estimate of the states from which (s, a) is easily reachable.

Given a sampled MDP M, Müller et al. (2022) argues that the model should estimate $\mathbb{E}_{M' \sim \mathbb{P}(M'|Z)}\left[\pi^*_{M'}(\cdot) \mid Z\right]$ to optimize for Bayesian inference.

We choose to directly regress against the optimal policy of the *sampled* MDP M. Concretely the target is its Boltzmann-optimal policy $\pi^*_{M,\tau} = \mathrm{softmax}(Q^*_M/\tau)$, a softened greedy policy that recovers the deterministic π^*_M as $\tau \to 0$ (cf. Appendix A). This is an unbiased estimate of the above quantity: since Z is generated by exploring M, the true M is already a posterior sample $M \sim P(M \mid Z)$, so regressing onto $\pi^*_{M,\tau}$ given Z is the same generative process as regressing onto $\pi^*_{M',\tau}$ for a fresh $M' \sim P(M \mid Z)$. Both yield the Bayes-optimal $\mathbb{E}[\pi^*_\tau(\cdot) \mid Z]$. Note that this is the optimal target in an *offline setting*, and the optimal exploration strategy may differ significantly. We observe empirically that such a target still delivers solid performance when used in an online setting.

Each training batch is a collection of freshly sampled MDPs. We draw every MDP from a broad prior that varies its size (from two to a few dozen states, with two to four actions), its connectivity, its transition stochasticity, and its reward structure. Connectivity is kept sparse: each state has a small outdegree, and its successors are usually the nearest neighbors in a latent geometry (a chain, a grid, a mesh, or an unstructured random graph) so that the transition graph has a coherent layout rather than arbitrary edges. Each transition row is drawn from a Dirichlet whose concentration spans near-deterministic to fully diffuse dynamics, so a single prior covers both clean and noisy environments. Rewards are sparse and assigned per state-action pair, with magnitudes and signs drawn independently, so per-step penalties and a sparse positive reward can coexist in the same MDP. Sparsity is applied explicitly: each (s, a) reward is independently zeroed with probability $1 - p_{\text{keep}}$, where p_{keep} is itself resampled for every MDP, so the fraction of nonzero rewards varies across the prior. We solve each sampled MDP exactly with value iteration to obtain its optimal policy, then simulate a finite budget of exploration to produce the noisy statistics Z the model actually sees, and train the network to recover the optimal policy from Z alone. The full distributions and constants are given in Appendix A.

4.2. Results

Although the model is trained with a purely offline objective, it can also be used to guide its own exploration autoregressively. This is the evaluation protocol we used in Figure 1 and Table 1. The full description of the evaluations is given in Appendix B.

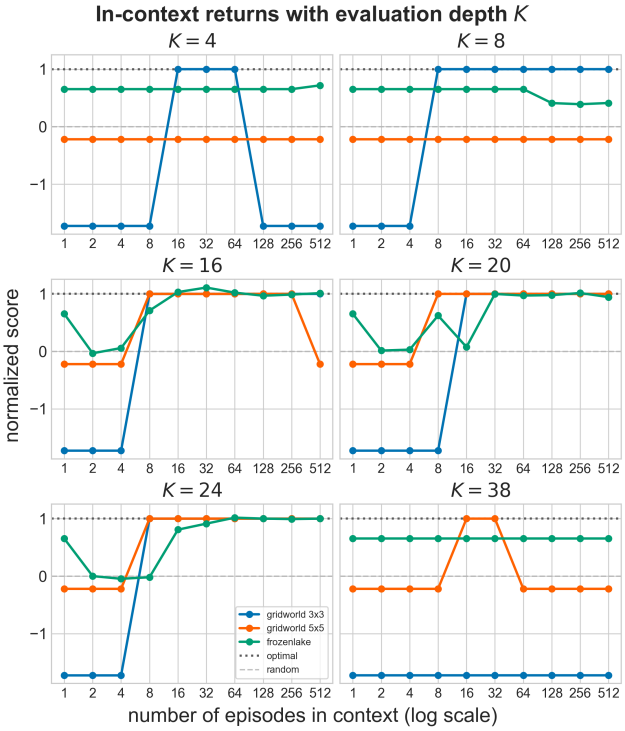

Figure 1. In-context returns as a function of the number of episodes in context, one panel per evaluation depth K and one line per held-out environment. The model was trained with $K = 20$. Since FrozenLake is a stochastic environment, the return is an average over 256 runs. Normalized scores are computed as $\frac{R - R_{\text{rand}}}{R_{\text{opt}} - R_{\text{rand}}}$ where R is the average model return, R_{opt} is the optimal average return and R_{rand} is the average return under a random policy.

The two benchmarks used here are GridWorld (Sutton & Barto, 2018) and FrozenLake (Brockman et al., 2016). GridWorld's deterministic environment is a $k \times k$ grid where each step taken has a negative rewards until the agent finds the self-absorbing goal case, which has a strong positive reward. FrozenLake is a highly stochastic analogue to GridWorld, with holes scattered across the grid and each action having a chance of making you "slip" to neighboring cases (which may be holes). Both holes and the goal are self-absorbing states, with strong positive and negative rewards respectively.

Our architecture allows for tunable depth at test time and we take advantage of that. Figure 1 shows that the model's performance degrades significantly as K diverges from 20, which is the value it was trained on. However, slight increases lead to improvement over the initial value, as is the case for $K = 24$ here. Consequently, we use $K = 24$ for the other evaluations.

Figure 2 isolates the model as a purely offline estimator. We

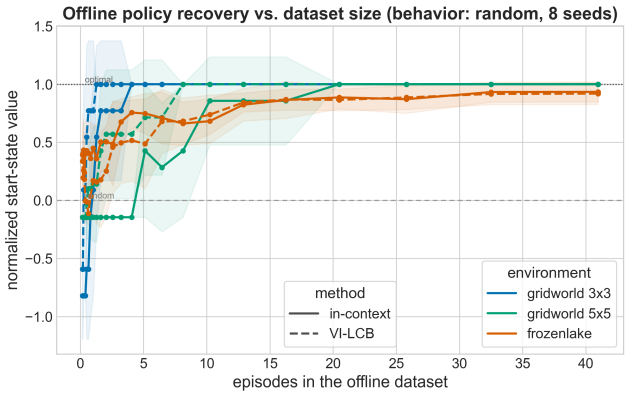

Figure 2. Offline policy recovery from a fixed uniform-random dataset. The figure plots the exact normalized start-state value of the policy recovered by the in-context model (solid) and by pessimistic value iteration (VI-LCB, dashed) against the number of episodes in the dataset, for three held-out MDPs (color); mean over 8 seeds (shaded: one standard deviation). The VI-LCB penalty coefficient is $c = 0.1$, hand-picked to maximize FrozenLake performance.

	In-Context	UCB-VI	Q-Learning
GridWorld 3x3	6	16	469
GridWorld 5x5	6	53	1205
FrozenLake	27	46	1014

Table 1. Median number of episodes it takes to converge, on 12 seeds.

compare our model to the VI-LCB offline RL algorithm (Rashidinejad et al., 2021), which performs pessimistic value iteration using lower-confidence-bound estimates. The fixed dataset is gathered by a uniform-random policy, both the model and VI-LCB must recover a policy from the same statistics, scored by the exact value at the start state. Both reach the optimum as the dataset grows, and on the deterministic GridWorlds a handful of episodes already suffice. The comparison is deliberately generous to the baseline, since the VI-LCB penalty was tuned on FrozenLake, yet the in-context model stays competitive throughout and recovers strong policies from very little data. FrozenLake is the hard case for both methods: under uniform exploration the goal sits behind the holes and is observed only rarely, so the dataset is coverage-limited and convergence is slower.

Table 1 shows that the model solves these problems with far fewer episodes than standard RL algorithms such as Q-learning (Watkins & Dayan, 1992), and even than a model-based algorithm such as UCB-VI (Azar et al., 2017), which is the upper-confidence-bound analogue of VI-LCB, often used for online RL.

Overall, it is pretty remarkable that the model generalizes to new unseen scenarios.

5. Limitations, Open Problems and Future Work

Our prototype is deliberately minimal, and several of its assumptions are also its limitations. We group them by the direction in which we expect them to be relaxed.

State features. The model is blind to the identity of a state by construction: the rows of Z are permutation-equivariant and carry no per-state features, so a state's integer index means nothing and cannot leak between MDPs. That buys invariance to relabeling at the cost of geometry. Two physically adjacent states look unrelated to the model until it observes a transition between them, and a state it has never visited is a uniform-prior blank with no neighbors to borrow from. One extension is to replace the bare index with a feature vector $x_s \in \mathbb{R}^d$ per state (and per action), which gives the input a notion of distance between states. The model could reach an unseen state by extrapolating from nearby, better-observed ones instead of leaning on the prior alone, the same way a tabular foundation model generalizes across feature space rather than row indices. This asks the prior to generate informative features whose distances track similarity in dynamics and value. Our generative process already places states in a latent geometry to wire up transitions (cf. Appendix A), and feeding that geometry to the model, not just the edges it induces, is a solid place to start.

Continuous states and actions. Enumerable (s, a) pairs are the core assumption of the tabular reduction, and large discrete or continuous state spaces break it. Feature-indexed rows point the way out on the state side: a dataset of observed (s, a, s', r) tuples keyed by features rather than indices is exactly the tabular shape our model already consumes. The action side needs a different head. Our masked softmax over a finite action set does not extend to continuous control; the head can instead emit the parameters of a squashed Gaussian, a Gaussian passed through a $\tanh$ to respect bounded action ranges, which is the standard continuous-action policy parameterization popularized by soft actor-critic (Haarnoja et al., 2018). This keeps our design choice of predicting a policy directly rather than action values.

Size generalization. Even with features, transfer across scales stays hard. A model trained on $|\mathcal{S}| \approx 50$ and deployed at $|\mathcal{S}| \approx 500$ meets graphs with different degree distributions, diameters and spectral gaps, so the dynamics that govern behaviors completely shift rather than merely grow. This is not a capacity problem; it likely needs an explicit scale prior in the data, or a representation invariant to size in a stronger sense than permutation equivariance.

Amortizing the online cost. Each environment step updates one row of Z and one count in $\mathcal{O}(1)$, already far cheaper than the $\Theta(NH)$ re-encoding of trajectory-based models. We currently re-encode all of Z to replan once per episode; whether the per-step update can instead be made incremental, refreshing only the affected rows of the propagation, is open and would make fully online deployment cheap.

Beyond the Markov statistic. The whole approach rests on the Markov property: Z is a sufficient statistic only when the environment is Markov in the observed state. Partial observability breaks this, and recovering a latent state would reintroduce the sequence-modeling costs we set out to avoid. Relatedly, the offline study (Figure 2) shows the method is only as good as the dataset's coverage: under poor exploration the prior carries the estimate, so a key open question is when the prior helps and when it misleads, i.e. how robust the method is to prior misspecification.

6. Conclusion

We have argued that prior design deserves to be a primary objective in in-context RL, and that the Markov property already provides the right representation for small problems: a fixed-size sufficient-statistics matrix that gives RL data a tabular shape amenable to non-sequential architectures. Open problems remain, but the broader foundation-model paradigm has taken hold in one domain after another by treating prior design as a first-class concern, and there are solid reasons to expect the same here.

Use of Generative AI

Generative AI was used to help with formatting and grammar checks.

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 of which we feel must be specifically highlighted here.

References

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 263–272, 2017.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.01345.

Choi, K., Cundy, C., Srivastava, S., and Ermon, S. LM-Priors: Pre-trained language models as task-specific priors. In *NeurIPS Workshop on Foundation Models for Decision Making*, 2022. URL https://arxiv.org/abs/2210.12530.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/1912.01588.

Dearden, R., Friedman, N., and Russell, S. Bayesian Q-learning. In *AAAI Conference on Artificial Intelligence*, 1998.

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. URL https://arxiv.org/abs/1611.02779.

Grigsby, J., Fan, L., and Zhu, Y. AMAGO: Scalable in-context reinforcement learning for adaptive agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.09971.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2207.01848.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körner, M., Hoo, R. S., Shen, H., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 2025. URL https://arxiv.org/abs/2501.02945.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., Gazeau, M., Sahni, H., Singh, S., and Mnih, V. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.14215.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.14892.

Lin, L., Bai, Y., and Mei, S. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.08566.

Müller, S., Hollmann, N., Pineda Arango, S., Grabocka, J., and Hutter, F. Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2112.10510.

Müller, S., Reuter, A., Hollmann, N., Rügamer, D., and Hutter, F. Position: The future of Bayesian prediction is prior-fitted. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2505.23947.

Qu, J., Holzmüller, D., Varoquaux, G., and Le Morvan, M. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.05564.

Qu, J., Holzmüller, D., Varoquaux, G., and Le Morvan, M. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. URL https://arxiv.org/abs/2602.11139.

Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., and Russell, S. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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. *Transactions on Machine Learning Research (TMLR)*, 2022. URL https://arxiv.org/abs/2205.06175.

Schiff, D., Lindenbaum, O., and Efroni, Y. Gradient free deep reinforcement learning with TabPFN. *arXiv preprint arXiv:2509.11259*, 2025. URL https://arxiv.org/abs/2509.11259.

Son, J., Lee, S., and Kim, G. Distilling reinforcement learning algorithms for in-context model-based planning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2502.19009.

Strens, M. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.03762.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

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.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Yan, X., Song, Y., Feng, X., Yang, M., Zhang, H., Ammar, H. B., and Wang, J. Efficient reinforcement learning with large language model priors. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07927.

Zabërgja, G., Kamel, R., Kadra, A., Frey, C. M. M., and Grabocka, J. End-to-end compression for tabular foundation models. *arXiv preprint arXiv:2602.05649*, 2026. URL https://arxiv.org/abs/2602.05649.

A. Implementation and training details

This appendix documents the proof-of-concept implementation: the prior over MDPs, the supervision targets, the model input, the architecture, and the optimization. The four stages map one-to-one onto the source files `prior.py`, `train.py`, `model.py` and `evaluation.py`.

A.1. Prior over MDPs

We sample finite MDPs $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with controlled variation in size, connectivity, transition stochasticity and reward structure; the discount is fixed at $\gamma = 0.95$. State and action counts are drawn per MDP,

$$\begin{aligned} S &\sim \mathrm{LogUniform}(2, 32) \\ A &\sim \mathrm{Uniform}(2, 4) \\ O &\sim \mathrm{PowerLaw}(o) \propto 1/o \end{aligned}$$

with an outdegree $O \in [\![1, 6]\!]$ shared across the states of a given MDP. Each MDP draws a latent geometry $g \in \{\text{chain}, \text{grid}, \text{mesh}, \text{random}\}$; states are placed as points $x_s \in [0, 1]^d$ ($d = 1, 2, 3$ for the geometric cases), and the successor set $\mathcal{N}(s, a)$ of size $OA_{\max}/2$ is taken as the nearest neighbors in this latent space, or uniformly at random when there is no geometry.

Transition rows are Dirichlet over a size-O support drawn from $\mathcal{N}(s, a)$,

$$\begin{aligned} P(\cdot \mid s, a) &\sim \mathrm{Dirichlet}(\alpha \mathbf{1}_O) \\ \alpha &\sim \mathrm{LogUniform}(0.05, 5.0), \end{aligned}$$

so the concentration α moves the dynamics between near-deterministic ($\alpha \ll 1$) and diffuse ($\alpha \gg 1$). Rewards are fixed per (s, a), sparsified by a per-MDP keep probability and signed independently per pair:

$$\begin{aligned} r(s, a) &= \mathbf{1}\{u_{s,a} < p_{\text{keep}}\}\, \sigma_{s,a}\, z_{s,a} \\ z_{s,a} &\sim \mathrm{Beta}(2, 5) \\ \Pr(\sigma_{s,a} = +1) &= p_{\text{pos}} \end{aligned}$$

with $p_{\text{keep}} \sim \mathrm{Beta}(2, 4)$ and the positive fraction $p_{\text{pos}} \sim \mathrm{Uniform}(0, 1)$ drawn once per MDP. Signing each pair separately, rather than fixing one sign per MDP, lets the prior cover mixed-sign reward structures, where per-step penalties and a sparse reward coexist in the same environment. The constants are listed in Table 2.

Table 2. Prior and model hyperparameters.

Symbol	Meaning	Distribution / value
γ	discount	0.95
S	states	$\mathrm{LogUniform}(2, 32)$
A	actions	$\mathrm{Uniform}(2, 4)$
O	outdegree	$\mathrm{PowerLaw},\ o \in [\![1, 6]\!]$
α	concentration	$\mathrm{LogUniform}(0.05, 5.0)$
g	latent geometry	uniform on geometries
p_{keep}	reward keep probability	$\mathrm{Beta}(2, 4)$
$z_{s,a}$	reward magnitude	$\mathrm{Beta}(2, 5)$
p_{pos}	positive-reward fraction	$\mathrm{Uniform}(0, 1)$
β	attention regularization	1.0

A.2. Supervision targets

For each sampled MDP we run value iteration on the true dynamics (500 sweeps) to obtain the optimal action value $Q^*(s, a)$, from which the supervision target is built. Both Q^* and the observed rewards are divided by the per-MDP reward scale $\max_{s,a} |r(s, a)|$, so the target is reward-scale invariant and the eval-time normalization (a running maximum) lines up with what the model saw in training.

A.3. Model input

The model never sees the true dynamics. At each step we simulate a finite exploration budget against (P, r): a per-MDP mean count is drawn log-uniformly and a Poisson number of visits is then sampled for each (s, a), so coverage spans the near-zero-data regime, where empirical rows fall back to the prior, up to well-sampled rows. A single feature builder, shared with evaluation, turns these statistics into the per-edge matrix Z whose rows hold the log visit count $\log(1 + N_{s,a})$, the normalized mean reward $\hat{r}_{s,a}$, and the empirical transition row $\hat{P}(\cdot \mid s, a)$ (uniform when the pair is unvisited). Because the same builder runs at training and at test time, the model's input distribution is identical in both.

A.4. Architecture

The model is a Graph Attention Network with hidden width d that plans over the MDP graph by iterating a weight-tied propagation layer. Each edge (s, a) is encoded from its scalar features, and the state and edge embeddings are initialized by pooling outgoing edges:

$$
\begin{aligned}
e_{sa} &= \phi_{\text{enc}}\big([\log(1 + N_{sa}),\, \hat{r}_{sa}]\big) \\
h_{sa}^{(0)} &= e_{sa} \\
h_{s}^{(0)} &= \phi_{\text{init}}\Big(\tfrac{1}{|\mathcal{A}|} \textstyle\sum_a e_{sa}\Big),
\end{aligned}
$$

where $\phi_{\text{enc}}, \phi_{\text{init}}$ are MLPs and $e_{sa} \in \mathbb{R}^d$. Note that $\hat{P}$ enters only as the attention bias below, not through ϕ_{enc}.

Each propagation step $\ell = 0, \ldots, K-1$ is a multi-head successor attention followed by two residual updates. With H heads of width $d_h = d/H$, the query of an edge (s, a) combines its current state-action embedding with its edge encoding, while keys and values come from the successor states s':

$$
\begin{aligned}
q_{sa}^{(i)} &= W_q^{(i)}[\, h_{sa}^{(\ell)};\, e_{sa}\,], \\
k_{s'}^{(i)} &= W_k^{(i)} h_{s'}^{(\ell)} \\
v_{s'}^{(i)} &= W_v^{(i)} h_{s'}^{(\ell)}
\end{aligned}
$$

Each (s, a) attends over candidate successors s' with logits that add the scaled dot product to the log empirical transition probability, and aggregates the corresponding values:

$$
\alpha_{sa}^{(i)}(s') = \frac{\langle q_{sa}^{(i)},\, k_{s'}^{(i)} \rangle}{\sqrt{d_h}} + \beta \log \hat{P}(s' \mid s, a) \tag{2}
$$

and then

$$
\begin{aligned}
m_{sa} &= W_o \ \text{concat}\left(m_{sa}^{(1)}, \ldots, m_{sa}^{(H)}\right) \\
m_{sa}^{(i)} &= \sum_{s'} \text{softmax}_{s'}\left(\alpha_{sa}^{(i)}\right) v_{s'}^{(i)}
\end{aligned}
$$

The additive bias in (2) is what ties the layer to planning, and its strength β acts as an attention regularizer that controls how tightly the attention is pinned to the empirical dynamics (Table 2; default $\beta = 1$). When the content logits $\langle q, k \rangle$ are constant in s', the weights reduce to $\text{softmax}_{s'}(\beta \log \hat{P}) \propto \hat{P}(\cdot \mid s, a)^{\beta}$: at $\beta = 1$ this is exactly $\hat{P}(\cdot \mid s, a)$, so the message collapses to the Bellman expectation $m_{sa} = \mathbb{E}_{s' \sim \hat{P}(\cdot \mid s,a)}[\, v_{s'}\,]$, while $\beta > 1$ sharpens and $\beta < 1$ flattens that distribution. The content term is a learned, data-dependent reweighting of this backup, and the $\log \hat{P} = -\infty$ entries suppress unreachable successors. The embeddings are then refreshed with residual MLPs and LayerNorm,

$$
\begin{aligned}
h_{sa}^{(\ell+1)} &= \text{LN}\big(h_{sa}^{(\ell)} + \phi_{sa}([\, h_{sa}^{(\ell)};\, m_{sa};\, e_{sa}\,])\big), \\
h_{s}^{(\ell+1)} &= \text{LN}\Big(h_{s}^{(\ell)} + \phi_{s}\big([\, h_{s}^{(\ell)};\, \tfrac{1}{|\mathcal{A}|} \textstyle\sum_a h_{sa}^{(\ell+1)}\,]\big)\Big).
\end{aligned}
$$

After K steps a readout produces a scalar logit per edge, and the policy is its masked softmax over the valid actions $\mathcal{A}(s)$:

$$
u_\theta(s, a) = \phi_{\text{head}}\big([\, h_s^{(K)};\, h_{sa}^{(K)};\, e_{sa}\,]\big), \qquad \pi_\theta(a \mid s) = \frac{\exp u_\theta(s, a)}{\sum_{a' \in \mathcal{A}(s)} \exp u_\theta(s, a')}.
$$

The head is read only through this softmax, so it outputs a policy rather than action values. Because the propagation layer is weight-tied across the K iterations, the network can be unrolled to a greater depth at evaluation than during training.

Padding and masking. MDPs in a batch vary in $|\mathcal{S}|$ and $|\mathcal{A}|$, so each is padded to the fixed maxima $S_{\max} = 32$ and $A_{\max} = 4$. A boolean state and action mask marks the valid rows: padded edges are zeroed in the encoder and after every propagation update, padded successors are dropped from the attention, and invalid actions receive a large negative logit before the policy softmax. Padding therefore never contributes to a valid row's representation, and the loss is averaged over valid states only.

A.5. Training objective and optimization

Each gradient step resamples a fresh batch of MDPs and their Z matrices. For a sampled MDP M, value iteration on the true dynamics gives Q^*, and the per-MDP scale $\rho = \max_{s,a} |r(s,a)|$ defines the temperature-τ Boltzmann target policy

$$\pi^*_\tau(a \mid s) = \mathrm{softmax}_{a \in \mathcal{A}(s)}\Big(\frac{Q^*(s,a)}{\rho\,\tau}\Big).$$

The objective is the expected per-state cross-entropy between this target and the model's prediction,

$$\mathcal{L}(\theta) = \mathbb{E}_{M,\, Z \sim \mathrm{explore}(M)}\left[\frac{1}{|\mathcal{S}|}\sum_{s \in \mathcal{S}}\Big(-\sum_{a \in \mathcal{A}(s)} \pi^*_\tau(a \mid s)\, \log \pi_\theta(a \mid s;\, Z)\Big)\right], \tag{3}$$

averaged over valid states and over the batch. Since cross-entropy with a soft target is minimized, for each fixed Z, at the conditional mean, the population minimizer of (3) is the posterior mean policy $\pi_\theta(\cdot \mid s;\, Z) = \mathbb{E}[\pi^*_\tau(\cdot \mid s) \mid Z]$, the Bayes-optimal estimate of Section 4. The KL between π^*_τ and π_θ is logged as a diagnostic only. We optimize with AdamW under a linear warmup followed by cosine decay with gradient-norm clipping. Default settings: hidden width 256, 8 attention heads, $K = 20$ propagation steps, dropout 0.05, batch size 128, $\tau = 0.2$, learning rate 3×10^{-4}, weight decay 0.01, and gradient clip 0.5.

B. Evaluation protocols

Both evaluations share the same autoregressive rollout. The model is never shown the true (P, r). It starts from a uniform transition estimate and zero rewards, and once per episode it re-plans a policy π_t from the statistics Z_t gathered so far. That policy is then held fixed for the episode while the experience it generates updates the counts that form Z_{t+1}. Every episode starts at $s_{\mathrm{start}} = 0$ and runs for 50 steps. Exploration samples actions from $\mathrm{softmax}(u_\theta/\tau)$ with temperature τ; the episodes we actually measure are run greedily ($\tau \to 0$, strict $\arg\max$) so that the recorded number reflects exploitation rather than exploration. The two protocols differ only in how a policy is scored.

B.1. Benchmarks

The two benchmarks used are GridWorld (Sutton & Barto, 2018) and FrozenLake (Brockman et al., 2016).

For GridWorld, the initial and final states are always respectively the top-left and bottom-right corners of the square. We set `step_cost` $= -1$ and `goal_r` $= 10.0$.

For FrozenLake, we use a 4×4 grid, with holes at index $5, 7, 11, 12$ (the grid is indexed row-major), `slip` $= 0.2$, `step_cost` $= 0$, `hole_r` $= -1$ and `goal_r` $= 1.0$.

B.2. Learning curves (Figure 1)

The first protocol measures the greedy return as a function of the number of episodes in context and of the evaluation depth K. We roll out 512 episodes per environment, built with $\tau = 0.3$ and read off the greedy return at every episode index that is a power of two, sweeping $K \in \{4, 8, 16, 20, 24, 38\}$ over the weight-tied propagation layer. A greedy episode is scored by its return R: the sum of rewards collected over the 50 steps, with no discounting. FrozenLake is stochastic, so a single rollout is noisy; we average 256 greedy rollouts per measured point and let only the first of them write to the statistics. The deterministic GridWorlds use a single rollout.

Returns are normalized against two references estimated by the same rollout, each averaged over 2000 episodes: R_{opt} from the value-iteration policy and R_{rand} from the uniform-random policy. The reported score is $(R - R_{\text{rand}})/(R_{\text{opt}} - R_{\text{rand}})$, so a random policy sits at 0 and the optimal policy at 1. Because both the numerator and R_{opt} are sampled, a near-optimal policy can land slightly above 1 on noise.

B.3. Episodes to convergence (Table 1)

The second protocol measures how many episodes the greedy policy needs before it is optimal, and compares that count against UCB-VI and tabular Q-learning. The score here is exact rather than sampled. At each episode we take the greedy policy $\pi_t = \arg\max_a u_\theta(s, a)$ and solve the linear system $V^{\pi_t} = (I - \gamma P_{\pi_t})^{-1} r_{\pi_t}$ for its discounted value at the start state. With V^* from value iteration and V_{rand} the exact value of the uniform-random policy, the normalized score is $(V^{\pi_t}(s_{\text{start}}) - V_{\text{rand}})/(V^*(s_{\text{start}}) - V_{\text{rand}})$. The greedy policy counts as converged once this score holds at or above the threshold for 8 consecutive episodes, so a single $\arg\max$ flip from an agent that is still exploring resets the count. Because the episodes we score are greedy regardless, we raise the exploration temperature to $\tau = 1.0$ when collecting data. The reported figure is the first episode of that window, taken as the median over 12 seeds. The two tabular baselines are scored the same way, episode by episode: UCB-VI plans on its empirical model plus a Hoeffding bonus with $c = 1$, and Q-learning is ε-greedy, with optimistic initialization, where $\varepsilon = 0.1$.

The exact discounted value is a stricter target than the sampled return of the figure. On the deterministic GridWorlds the model recovers the optimal policy outright, so both protocols agree and the score is exactly 1. On FrozenLake the in-context policy settles just under the optimum in exact value. The table is therefore sensitive to the threshold: at 0.95 it converges within tens of episodes, while a threshold near 1 would never register, even though the sampled curve in the figure already looks saturated.

B.4. Offline policy recovery (Figure 2)

The third protocol isolates the model as an offline estimator: given a fixed dataset of transitions, how good a policy can it recover, and how does that compare against a standard offline-RL baseline on the identical data. We fix the behavior policy to uniform random, reset to $s_{\text{start}} = 0$ whenever an absorbing state is reached or after 50 steps, and collect a single stream of transitions per seed. At every dataset size that is a power of two (from 8 to 2048 transitions) we freeze the accumulated statistics, the counts $N(s, a, s')$, the summed rewards, and the observed reward scale, and feed the identical Z to both estimators. Each produces a policy, scored by the exact discounted normalized start value of the previous protocol. The curves are the mean over 8 seeds with one standard deviation shaded.

The baseline is pessimistic value iteration (VI-LCB, Rashidinejad et al., 2021), the offline counterpart of UCB-VI. It plans on the maximum-likelihood model built from the counts, subtracting a Hoeffding penalty $c\sqrt{\log(|\mathcal{S}||\mathcal{A}|/\delta)/N(s, a)}$ from each reward so that under-visited pairs are discouraged; pairs never seen in the data are pinned to the pessimistic value floor $-r_{\text{scale}}/(1 - \gamma)$, and absorbing states are given continuation value 0, since their termination is recorded in the dataset. The in-context model sees the same Z and acts greedily.

We hand-picked c to maximize performance on FrozenLake. A sweep over $c \in \{0, 0.1, 0.3, 0.5, 1, 2\}$ left the deterministic GridWorlds at the optimum for every value, but changed FrozenLake sharply: there the penalty over-suppresses the rare observed goal transitions, so a large c drives the recovered policy below random. We report $c = 0.1$, the value that maximizes the FrozenLake offline score. FrozenLake is the hard case for any data-only method here: under uniform exploration from a fixed start the goal sits behind the holes and is observed only a handful of times in thousands of transitions, so VI-LCB is coverage-limited while the model can fall back on its prior.