

DenseMem Cross Level Credit Assignment over Hierarchical Memory for Long Horizon Agentic Decision Making

Zitan Chen

zitanc@alumni.cmu.edu

Carnegie Mellon University

Xiang Li

Georgia Institute of Technology

Yue Ning

University of Toronto

Wentao Zhang

University of Southern California <https://orcid.org/0009-0009-5884-7961>

Research Article

Keywords: hierarchical memory, credit assignment, sparse rewards, episodic control, reinforcement learning

Posted Date: October 19th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11395802/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

DenseMem Cross Level Credit Assignment over Hierarchical Memory for Long Horizon Agentic Decision Making

Zitan Chen^{1,*} Xiang Li² Yue Ning³ Wentao Zhang⁴

¹Carnegie Mellon University, Pittsburgh, PA 15213, USA ²Georgia Institute of Technology, Atlanta, GA 30332, USA

³University of Toronto, Toronto, ON M5S 1A1, Canada ⁴University of Southern California, Los Angeles, CA 90089, USA

*zitanc@alumni.cmu.edu xli890@gatech.edu lydianing59@gmail.com wentao@alumni.usc.edu

Abstract—DenseMem combines bounded episodic memory with signed credit allocation across episodes, fixed-length segments, and transitions. We ask whether this allocation improves delayed-reward learning beyond retrieval alone under equal interaction and update budgets. We formalize the method, characterize restricted policy-invariance conditions, and otherwise identify biased reward shaping. Five-seed experiments yield mixed results: Taxi success is 87.0%, above flat credit at 58.7% but below unshaped learning at 100.0%; MiniGrid 16 reaches 99.7% without improving route efficiency. The contribution is an auditable allocation mechanism and a controlled account of its limitations, rather than evidence of general long-horizon superiority.

Keywords—hierarchical memory; credit assignment; sparse rewards; episodic control; reinforcement learning.

I Introduction

Delayed rewards obscure intermediate contributions. Q-learning propagates feedback through value backups [1]; neural episodic control retrieves stored values [2]. DenseMem specifies how stored outcomes modify local learning signals.

Research question. With the learner, observations, memory capacity, interactions, and backups held fixed, does signed episode-to-segment-to-transition allocation improve unshaped test success and return over hierarchical retrieval alone and a flat-credit variant? A separate theoretical question is when the added credit preserves the task objective.

The mechanism uses exact-match historical outcomes and nested softmax normalization, without process labels, a learned encoder, or executable subpolicies. Its contribution is a bounded memory estimator, an explicit allocation rule with a verifiable sum identity, and matched-budget ablations. The primary comparison is DenseMem versus Hierarchy; the flat-credit comparison changes both retrieval and allocation and therefore is not a pure test of segmentation.

The evaluation concerns discrete navigation and transport tasks. It does not establish improvement for language agents or inherently long planning sequences.

II Related Work

Table I contrasts retrieval, temporal abstraction, and credit allocation. Recent methods use demonstration alignment, distributional baselines, model-based chunking, or segment advantages; DenseMem instead uses empirical outcome differences and nested normalization.

TABLE I. Mechanistic comparison with related work.

Approach	Representation and signal	Difference from DenseMem
Episodic control [2]	Stored values with learned-key retrieval.	Retrieval without signed nested credit.
ReAct / Reflexion [3], [4]	Reason-act traces / reflective text memory.	Language feedback, not numerical credit allocation.
Options [5]	Temporally extended policies.	Execution hierarchy, not fixed credit blocks.
RUDDER / Align-RUDDER [6], [7]	Return prediction / aligned demonstrations for redistribution.	Redistributes rewards; DenseMem adds credit.
HCA / QCA [8], [9]	Outcome-conditioned likelihoods / quantile-based baselines.	Likelihood or distributional estimates, not key differences.
Chunked-TD [10]	Model-predicted transitions set multistep targets.	Adapts TD targets, not additive reward credit.
SPO / HCAPO [11], [12]	Monte Carlo segment advantages / LLM hindsight critic.	Policy optimization, not tabular nested allocation.
DenseMem	FIFO outcomes; fixed blocks; signed nested weights.	Explicit sum identity; restricted invariance.

DenseMem uses empirical associations, not causal effects. Return equivalence in RUDDER and policy invariance under potential-based shaping [13] do not extend to arbitrary additive credit. Options execute subpolicies; DenseMem’s fixed blocks only organize allocation.

III Algorithm and Implementation

A Notation and bounded memory

Write transition t as $e_t = (s_t, a_t, s_{t+1}, r_t)$, rather than τ_t , to reserve τ exclusively for temperature. An episode E has T transitions, undiscounted task return $R = \sum r_t$, and discounted return $G_\gamma = \sum \gamma^t r_t$. Temperature $\tau > 0$ controls allocation concentration; discount $\gamma \in [0, 1]$ weights future rewards; learning rate $\alpha \in (0, 1]$ controls each Q update. These quantities have distinct roles.

Memory B (Fig. 1) retains $M = 64$ completed episodes, including horizon failures, in FIFO order. Each episode links to $J = \lceil T/L \rceil$ blocks of $L = 8$ transitions. For $j = 0, \dots, J-1$, block j covers $I_j = \{u_j, \dots, v_j - 1\}$, with $u_j = jL$ and $v_j = \min((j+1)L, T)$. A final block may be short. Checkpoint-censored fragments have no outcome label.

B Outcome estimates and signed allocation

Let $k_0(s) = s$ and $k_1(s) = g(s)$. For level $\ell \in \{0, 1\}$, $N\ell(x)$ counts buffered episodes containing key x and $Y\ell(x)$ sums their R values. Deduplicate local states including s_T , coarse states at all block boundaries, local state-action pairs, and coarse pairs $(g(s_{u_j}), a_{u_j})$, separately within each episode. Pair statistics are $N\ell(x, a)$, $Y\ell(x, a)$. With b the mean buffered return (zero when empty), define

$$V_\ell(x) = \frac{Y_\ell(x) + b}{N_\ell(x) + 1} \quad (1)$$

$$U_\ell(x, a) = \frac{Y_\ell(x, a) + V_\ell(x)}{N_\ell(x, a) + 1} \quad (2)$$

The centered score $D\ell(x, a) = U\ell(x, a) - V\ell(x)$ is zero for an unseen pair. Hierarchical retrieval is

$$h(s, a) = \frac{1}{2}[D_0(s, a) + D_1(g(s), a)] \quad (3)$$

Here β scales retrieval, κ scales added credit, and λ scales the local difference term. Collect with ϵ -greedy scores $Q(s, a) + \beta h(s, a)$, breaking ties uniformly. Freeze B before crediting the completed episode; it must not contribute to its own estimates. Let $A = R - b$ and $z = \text{sign}(A)$. For adjacent states and block endpoints,

$$\Delta_t^0 = V_0(s_{t+1}) - V_0(s_t) \quad (4)$$

$$\Delta_j^1 = V_1(g(s_{v_j})) - V_1(g(s_{u_j})) \quad (5)$$

$$w_j = \frac{\exp(z\Delta_j^1/\tau)}{\sum_k \exp(z\Delta_k^1/\tau)} \quad (6)$$

$$w_{t|j} = \frac{\exp(z\Delta_t^0/\tau)}{\sum_i \in I_j \exp(z\Delta_i^0/\tau)} \quad (7)$$

$$c_t = Aw_j w_{t|j} + \lambda \Delta_t^0, \quad r'_t = r_t + \kappa c_t \quad (8)$$

Equations (6) and (7) normalize over all blocks and within block j , respectively. Implement softmax as $\exp(x - \max x)/\sum \exp(x - \max x)$; reject nonfinite inputs and $\tau \leq 0$. Positive A emphasizes increases in estimated outcome; negative A emphasizes decreases and assigns negative credit. Only $A w_j w_{t|j}$ must share A 's sign: $\lambda \Delta_t^0$ can reverse an individual credit. If $A = 0$, the allocation vanishes but the local term can remain.

Both normalizations sum to one, and local differences telescope, giving the exact identity

$$\sum_t c_t = A + \lambda[V_0(s_T) - V_0(s_0)] \quad (9)$$

As $\tau \rightarrow \infty$, credit mass is A/J per block and $A/(J|I_j|)$ per transition when $\lambda = 0$. Thus a short final block receives greater per-step mass. $L \geq T$ gives one local softmax; $L = 1$ gives coarse weights over individual steps.

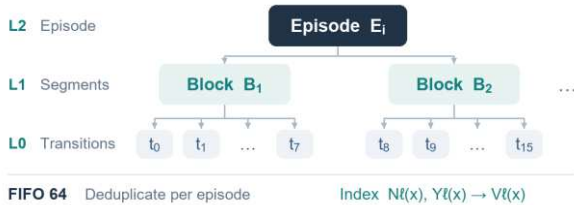


Fig. 1. Episode-block-transition memory. Deduplicated keys update the outcome index once per episode.

C Policy invariance and bias

A reward transformation preserves an objective only if it preserves policy ordering under that objective. Equation (9) conserves advantage when $\lambda = 0$; it does not conserve the original discounted return. The implemented $\gamma = 0.99$, $\lambda = 0.1$ rule is biased reward shaping, with no general optimal-policy guarantee.

Restricted sufficient condition for the existing rule. For finite episodes, $\gamma = 1$, $\lambda = 0$, fixed policy-independent b , and $1 + \kappa > 0$, summing (8) yields

$$G' = (1 + \kappa)R - \kappa b \quad (10)$$

For a fixed initial-state distribution this positive affine transformation preserves expected-return ordering over complete policies, regardless of allocation weights. The same conclusion holds with $\lambda \neq 0$ if the expected endpoint term in (9) is policy-independent. These are objective-level statements, not convergence guarantees for a history-dependent tabular update. A moving, policy-dependent memory baseline does not establish these assumptions.

Standard alternative. For a Markov task state x_t (including elapsed time for a finite horizon), replace the entire added reward by a fixed bounded potential difference [13]:

$$r'_t = r_t + \gamma \Phi(x_{t+1}) - \Phi(x_t) \quad (11)$$

The discounted shaping sum is $-\Phi(x_0) + \gamma^T \Phi(x_T)$. With zero terminal potential, or a vanishing tail in the discounted infinite-horizon case, only a policy-independent initial offset remains. This preserves the optimal policy of the reward objective. Replacing only $\lambda \Delta_t^0$ leaves the trajectory-dependent allocation term in (8), so it is insufficient.

Why discounting matters. Consider equal two-step trajectories in one block, with terminal reward 1, $b = 0.5$, $\lambda = 0$, and $\gamma = 0.9$. Feasible softmax weights (0.9, 0.1) and (0.1, 0.9) both sum to one, yet their shaped returns are $0.9 + 0.495\kappa$ and $0.9 + 0.455\kappa$. Thus $\kappa > 0$ breaks an original tie; the sum identity alone cannot preserve the set of optimal policies.

Bias direction. For positive κ , earlier positive credit has more discounted influence; earlier negative credit incurs a larger penalty. Signed weights favor transitions consistent with historically good or bad outcomes, which can reinforce incidental detours or punish necessary moves when keys alias distinct contexts. Uniform block mass further overweights short blocks per transition. The undiscounted local difference also rewards rises in empirical outcome estimates without the discounted cancellation required by (11). These are structural tendencies, not established causes of the observed failures.

Reward invariance also does not ensure invariance of the deployed action rule: βh can change $\text{argmax } Q$. A guarantee needs $\beta = 0$ or a retrieval term that preserves the maximizing action set. Constant α and time-omitting states provide no standard tabular convergence guarantee here.

D Complete training procedure

Algorithm 1. DenseMem with one backup per interaction.

Input: task wrapper, frozen reset splits, budget N , checkpoints C , M , L , α , γ , τ , λ , κ , β , and training seed σ .

- 1 Set $Q = 0$, $B = \emptyset$, all counts/sums $= 0$, $b = 0$, $n = 0$.
- 2 Seed action RNG with σ ; reset scheduler with 100000 + σ .
- 3 While $n < N$:
- 4 $c \star \leftarrow$ next checkpoint; sample a training reset uniformly.
- 5 Reset environment; $E \leftarrow$ empty; done $\leftarrow$ false.
- 6 While not done and $n < c \star$:
- 7 $\epsilon \leftarrow \max(0.1, 1 - 0.9n/(0.8N))$.
- 8 Sample uniform action with probability ϵ ; otherwise choose uniformly among $\arg\max_a [Q(s,a) + \beta h(s,a)]$.
- 9 Step wrapper; append e_t , native reward and end flags; $n \leftarrow n + 1$; done $\leftarrow$ success or environment horizon.
- 10 If done: compute R and A from the frozen B ; partition E ; compute (1)–(8). Otherwise set all $c_t = 0$.
- 11 For $t = T - 1, \dots, 0$, perform exactly one update (12).
- 12 If done: insert E and its deduplicated keys/pairs into B ; increment N, Y once per key/pair by 1; R ; if $|B| > M$, evict oldest E and subtract its contributions.
- 13 Recompute b , V , U and both retrieval caches.
- 14 If $n = c \star$: freeze Q, B ; evaluate every held-out reset with $\epsilon = 0$ and tie RNG seeded 700000 + σ ; restore training RNG; advance to next checkpoint.
- 15 Start a new training reset; never resume a censored fragment.

Output: final Q and retrieval caches; logs and configuration hash.

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[r_t' + \gamma m_t] \quad (12)$$

Here $m_t = 0$ only at the final transition of a completed episode (success or task horizon); otherwise $m_t = \max_a Q(s_{t+1}, a)$, using the current Q during the backward sweep. A checkpoint-censored fragment uses raw rewards and bootstraps at its last observation. Empty fragments are not updated. Evaluation changes neither Q nor B .

Base uses Q and raw rewards; Flat memory uses $Q + \beta D_0$ and raw rewards; Hierarchy uses $Q + \beta h$ and raw rewards. DenseMem adds (8). Flat credit uses local retrieval and a single softmax over all $z \Delta_i^0 / \tau$, with the same A , τ , λ , κ , and capacity. All methods use identical reverse backups; none adds replay. Figure 2 illustrates the allocation.

Allocation costs $O(T + J)$; backups cost $O(T|\mathcal{A}|)$. Episode storage is $O(MH)$ for horizon H . The implementation uses dense count and cache arrays, costing $O((|\mathcal{S}| + |\mathcal{G}|)|\mathcal{A}|)$ memory and time per full cache refresh, plus episode deduplication. The linear allocation claim excludes these costs and is not a latency measurement.

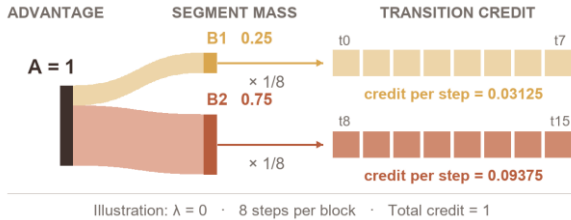


Fig. 2. Illustrative allocation, not measured data: $A = 1$, $\lambda = 0$, block weights 0.25/0.75, and uniform local weights 1/8. Per-step credits are 0.03125/0.09375.

IV Experimental Evaluation

A Observation keys and reward adaptation

Taxi-v3 (Gymnasium 1.2.3 [14]) uses all six actions without the action mask. Decode integer s into row r , column c , passenger p , destination d ; the exact coarse index is $g(s) = (((\lfloor r/2 \rfloor \cdot 3 + \lfloor c/2 \rfloor) \cdot 5 + p) \cdot 4 + d)$. It merges spatial cells but retains passenger and destination. The 500-state table maps to 180 coarse indices.

MiniGrid 3.0.0 [15] wraps `EmptyEnv(size = n, agent_start_pos = None, max_steps = H)` with `FullyObsWrapper`. From `image[x,y]`, read the agent position and direction d ; $s = 4(n_x + y) + d$. Set $q_x = \min(1, \lfloor x/(n/2) \rfloor)$, likewise q_y , and $g(s) = 4(2q_x + q_y) + d$. This yields 16 coarse indices. Actions are left, right, and forward; elapsed time is omitted from the learned state.

Taxi’s native step, illegal-action, and delivery rewards are logged but replaced for learning and primary evaluation by 1 for successful delivery and 0 otherwise, with $H = 256$. This changes the task objective and removes native penalties; comparisons are within this adaptation. MiniGrid retains native success reward $1 - 0.9T/H$ and zero failure reward, with $(n, H) = (8, 256), (12, 576), (16, 1024)$. “Unshaped return” means these wrapper rewards before (8), not necessarily the native benchmark return.

B Reset enumeration and selection

For each task, enumerate reset seeds 0,...,99999 in ascending order; retain the first seed for each encoded initial state. Stop at 300 unique Taxi starts or $4[(n - 2)^2 - 1]$ grid starts (the occupied goal is excluded). Sort by state, shuffle once with NumPy default_rng(20260911), and split at [0.6K] and [0.8K]. The saved reset seeds are the task identifiers’ representatives.

Train/validation/test sizes are 180/60/60 for Taxi and 84/28/28, 237/79/80, 468/156/156 for grids 8/12/16. Training draws starts uniformly with replacement. Held-out starts can be visited during training; this tests new initial tasks on known maps, not unseen states or maps. Evaluation asserts reset-state identity.

C Hyperparameters and matched budgets

Seeds 11, 22, 33, 44, 55 each receive 60,000 interactions and backups. Fixed choices are $\alpha = 0.2$, $\gamma = 0.99$, $M = 64$, $L = 8$, $\tau = 0.25$, $\lambda = 0.1$; these were not individually tuned. Exploration follows Algorithm 1. Validate the Cartesian grid $\kappa \in \{0.02, 0.1, 0.2\}$, $\beta \in \{0.005, 0.05\}$ on DenseMem, four tasks and seeds 7, 8 at 20,000 steps.

Maximize the unweighted mean of eight per-task/seed validation returns; ties favor smaller κ then β . The chosen (0.2, 0.005) scores 0.72407. Reuse this pair across memory methods; there is no per-baseline tuning. Validation uses 48 runs (960,000 interactions) in addition to 100 final runs (six million). The test set does not select parameters or checkpoints.

D Checkpoints and reported results

Stop training exactly at 20k, 40k, and 60k interactions. If a checkpoint interrupts an unfinished episode, use raw-reward backups, retain a bootstrap target, skip memory insertion, then reset after evaluation. A coincident success/horizon remains complete. The prespecified 60k model is primary; there is no best-test-checkpoint selection. Saved NPZ files contain Q and local/hierarchical caches for inference, not the FIFO/RNG state needed for exact training resumption.

Results are mean $\pm$ sample SD over five seeds; steps include failures and return is unshaped. Repeated starts and checkpoints are not independent seeds.

TABLE II. Final test performance at 60,000 steps.

Condition	Success %	Return	Steps
Taxi H = 256			
Base	100.0 $\pm$ 0.0	1.000 $\pm$.000	14.3 $\pm$ 0.6
Flat memory	100.0 $\pm$ 0.0	1.000 $\pm$.000	14.2 $\pm$ 0.4
Hierarchy	100.0 $\pm$ 0.0	1.000 $\pm$.000	14.3 $\pm$ 0.6
DenseMem	87.0 $\pm$ 12.7	.870 $\pm$.127	45.5 $\pm$ 30.4
MiniGrid 16 H = 1024			
Base	100.0 $\pm$ 0.0	.985 $\pm$.000	16.9 $\pm$ 0.3
Flat memory	100.0 $\pm$ 0.0	.985 $\pm$.000	16.7 $\pm$ 0.2
Hierarchy	99.7 $\pm$ 0.6	.982 $\pm$.006	19.7 $\pm$ 6.1
DenseMem	99.7 $\pm$ 0.6	.981 $\pm$.006	21.2 $\pm$ 6.2

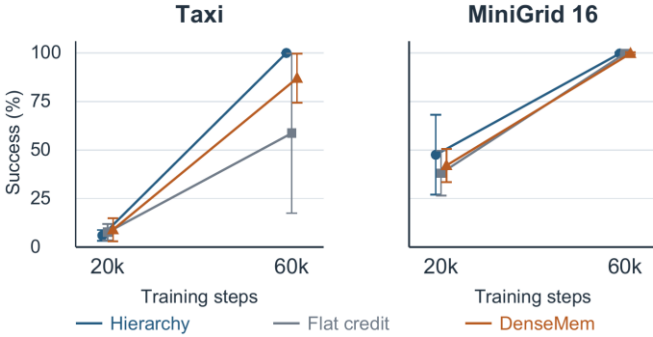


Fig. 3. Success at two prespecified checkpoints (five-seed mean $\pm$ sample SD). Slight horizontal offsets separate overlapping markers; connecting lines do not show intervening learning progress.

DenseMem trails Base by 13.0 percentage points on Taxi and uses about 3.18 times as many steps per episode (45.5/14.3, including failures). On MiniGrid 16, its return is lower by 0.004 and its mean episode length is approximately 25% greater. These results provide no final control-efficiency advantage.

In Fig. 3, final Taxi success exceeds Flat credit by 28.3 points, but removing shaping while retaining hierarchical retrieval raises success to 100%. The primary comparison does not support a shaping benefit. At 20k on MiniGrid 16, DenseMem is 5.5 points below Hierarchy. These snapshots do not establish faster learning or statistical significance.

E Limits and reproducibility

DenseMem’s grid 8/12/16 success is 30.7%/100.0%/99.7%; grid 8 SD is 45.1 points. This nonmonotonic result does not establish scalability. Cycles occur in saved trajectories, including a Base run. Time-omitting states, constant learning rate, and reward bias are hypotheses requiring separate controls; horizon caps are not solution lengths.

To isolate segmentation, keep hierarchical retrieval and replace only nested weights by one transition softmax. Separately test $\lambda = 0$, uniform weights, sign reversal, time-augmented states, and fixed-map reward delays; sweep L and M on validation tasks. These are proposed experiments, not measured results.

Frozen source/configuration hashes match the archived manifest; reaggregation verifies 100 runs, six million training transitions, and 24,300 test episodes across three checkpoints. Logs retain native/task rewards, failures, and censored fragments. Reruns require saved splits, Python 3.12.14 and NumPy 2.5.3, in addition to the environment versions above. The runs used an i7-14650HX CPU and eight workers. Five seeds and one tabular learner do not establish language-agent performance.

V Conclusion

DenseMem makes hierarchical attribution explicit and auditable. The sum identity has restricted objective-preservation conditions; the evaluated configuration remains biased shaping. Equal-budget evidence favors DenseMem over flat credit on Taxi, but not over unshaped control. The open question is whether isolated allocation changes improve learning under controlled reward delays.

References

- [1] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, pp. 279–292, 1992. doi: 10.1007/BF00992698.
- [2] A. Pritzel et al., “Neural episodic control,” *Proc. ICML*, vol. 70, pp. 2827–2836, 2017.
- [3] S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” *Proc. ICLR*, 2023.
- [4] N. Shinn et al., “Reflexion: Language agents with verbal reinforcement learning,” *NeurIPS*, vol. 36, pp. 8634–8652, 2023.
- [5] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.
- [6] J. A. Arjona-Medina et al., “RUDDER: Return decomposition for delayed rewards,” *NeurIPS*, vol. 32, 2019.
- [7] V. Patil et al., “Align-RUDDER: Learning from few demonstrations by reward redistribution,” *Proc. ICML*, vol. 162, pp. 17531–17572, 2022.
- [8] A. Harutyunyan et al., “Hindsight credit assignment,” *NeurIPS*, vol. 32, 2019.
- [9] T. Mesnard et al., “Quantile credit assignment,” *Proc. ICML*, vol. 202, pp. 24517–24531, 2023.
- [10] A. Ramesh et al., “Sequence compression speeds up credit assignment in reinforcement learning,” *Proc. ICML*, vol. 235, pp. 42104–42126, 2024.
- [11] Y. Guo et al., “Segment policy optimization: Effective segment-level credit assignment in RL for large language models,” *NeurIPS*, vol. 38, pp. 114399–114431, 2025.
- [12] H.-Z. Tan et al., “Hindsight credit assignment for long-horizon LLM agents,” *arXiv preprint arXiv:2603.08754*, 2026.
- [13] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” *Proc. ICML*, pp. 278–287, 1999.
- [14] M. Towers et al., “Gymnasium: A standard interface for reinforcement learning environments,” *NeurIPS*, vol. 38, 2025. doi: 10.52202/085713-4916.
- [15] M. Chevalier-Boisvert et al., “Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks,” *NeurIPS*, vol. 36, pp. 73383–73394, 2023.