MiniGrid DoorKey — Midterm Report

Hubert Yakpe Trolindo · CAP6671, Fall 2026 · repository **ucf-cap6671/fall2026-doorkey-cybsecoff31** · baselines at commit **97c4f89**; comparison runs at **55d359a** (G1, G2) and **5caf2f9** (G3) · October 2026

1 Baseline replication

In the following table, I compare the unchanged PPO baseline against a uniform-random policy on the same layouts. Both are scored under the course protocol: 200 episodes, evaluation seed 12345 (episode *i* on layout seed 12345+*i*), sampled actions, success = return > 0. All values are from each run's `eval.json`.

Grid	Budget	Trained success	Trained return	Trained length	Random success	Random return	Random length	Run directory (runs/…)
5×5	500k	**100%**	0.959	11.5	**6.0%**	0.025	244.7	`baseline_doorkey5x5_seed1`
6×6	1.0M	**100%**	0.968	12.9	**4.0%**	0.017	354.9	`baseline_doorkey6x6_seed1`
8×8	1.0M	**0%**	0.000	640.0	**1.5%**	0.007	636.4	`baseline_doorkey8x8_seed1`
16×16	1.0M	**0%**	0.000	2560.0	**1.5%**	0.005	2549.7	`baseline_doorkey16x16_seed1`

Tab. 1 — Baseline replication: trained PPO vs. untrained random policy (eval.json).

What breaks down, and where? For the PPO baseline, success rate and mean return are the quantities that break down, and both collapse at **8×8**, falling from 100% / ≈0.96 at 5×5 and 6×6 to 0% / ≈0.0 at 8×8 and 16×16. Mean episode length confirms this: it rises to the environment's truncation cap (640 steps at 8×8 and 2,560 at 16×16), so the trained agent times out in every evaluation episode. At these sizes the trained policy performs no better than a uniform-random policy (0% vs. 1.5% success), which indicates a genuine learning failure rather than a minor regression.

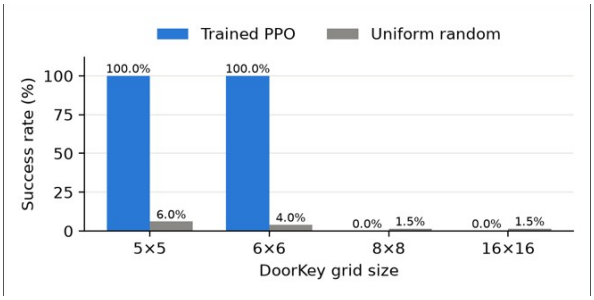


Fig. 1 — Protocol success rate by grid size, trained vs. random (eval.json).

2 Failure analysis on 8×8

Description. The baseline agent wanders until the 640-step cap (`length_mean` in `runs/baseline_doorkey8x8_seed1/eval.json`) and completes only 1,561 episodes across the whole 1M steps (rows of `episodes.csv`): 99.6% of these episodes hit the episode cap (only 6 ended before 640 steps), and evaluation gives length 640.0 on all 200 episodes. How does the agent fail? The five recorded evaluation episodes per policy (`render/sampled`, `render/random`; same layouts) can be decoded frame by frame, extracting the agent cell, whether the key is still on the floor, and the door state. The trained agent **picks up the key in 5/5 episodes but never unlocks the door (0/5)** and never passes it. Its walk even shrinks: it covers 6 to 12 cells (mean 8.6) and spends a mean 42% of all 640 steps on a single cell, right next to the key's original position. By comparison, the random policy covers 15 to 20 cells (mean 17.2), spends at most 14 to 19% of steps on any one cell, picks up the key in 4/5 episodes and unlocks the door once (Fig. 2). The sample is small, but all five trained episodes show the same pattern.



Fig. 2 — Share of the 640 evaluation steps spent in each cell (K key, D door, G goal), from the recorded GIFs of evaluation layouts 0 and 3.

Where the agent's training stalls. `runs/baseline_doorkey8x8_seed1/episodes.csv` shows that the agent succeeds only 6 times in 1M steps, all before step 202k (Fig. 5). These lucky random walks were never reinforced: there are **zero** successes over the final ≈800k steps, where the random rate predicts ≈19 and the early rate ≈24. Zero is a $\sim 10^{-8}$ event, meaning the agent became *worse* than random rather than staying random. For comparison, 6×6 first exceeds a mean return of 0.5 at step 289,792 and stays above 0.9 from step 366,592. The high entropy at 8×8 (Fig. 4), about 1.9 ≈ ln 7 at the start of training (all 7 actions picked with nearly equal probability), underlines that the early successes were luck rather than genuine learning. Entropy falls to 0.74 at step 315k with no reward to justify the commitment, then climbs back and stays high to the end of the run (1.66 vs. 0.02 at 6×6).

Moreover, the final explained variance of −2.5 (0.81 at 6×6) in `runs/baseline_doorkey8x8_seed1/metrics.csv` confirms that the learning signal from the occasional success was weak, noisy, and forgotten before the next one.

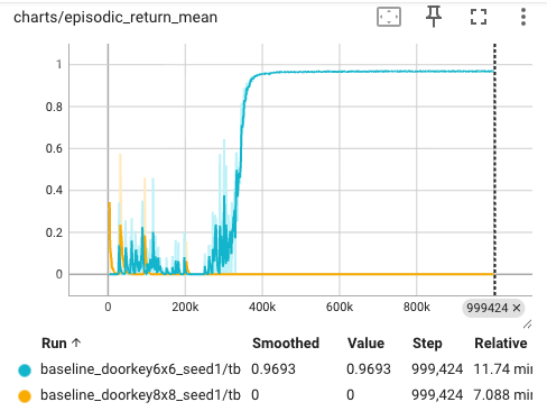


Fig. 3 — PPO training return. 6×6 breaks through at ≈290k steps; 8×8 flatlines.

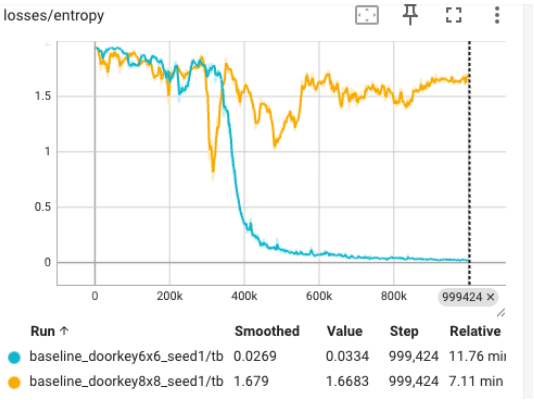


Fig. 4 — Policy entropy. 6×6 collapses to ≈0.03; 8×8 remains high (≈1.67).

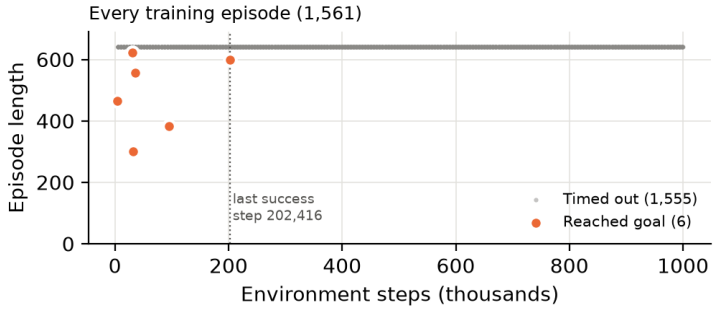


Fig. 5 — Every 8×8 training episode: length and the six successes (episodes.csv).

Explanation. The key takeaway is that the agent is experiencing a sparse-reward credit-assignment failure compounded by partial observability. The bottleneck sits at the door, not at the key pick-up. To reach the goal the agent needs key → toggle-door → walk to goal. Key pick-up happens even under random play, but unlocking the door and then finding the goal are rare. The critic (*the value function*, which predicts the future reward the agent can expect from state *s*) learns state values mostly from its own predictions, and real rewards are the only anchor. Six reward events are clearly not enough to train it. With γ (the discount factor) = 0.99 and the GAE λ = 0.95 (`conf/ppo/default.yaml`), a reward's credit shrinks by a factor of 0.94 at each step back, halving about every 11 steps. Each success mainly teaches the ~20–50 states just before the goal, is used for one on-policy batch and discarded, and the states it informs lie past the door, where the agent almost never goes. The actor (*the policy*, the other half of the agent, which decides what to do in the situations the critic judges good) does not learn from raw reward but from the advantage, so if the critic cannot make meaningful predictions, the policy has nothing meaningful to learn from. These uninformative values make the per-minibatch-normalized advantages noise; the policy keeps high entropy over mostly non-moving actions and settles into a small spatial loop beside the key. Partial observability adds to this without being an absolute barrier: in the left room (≤ 3 cells wide) the door is inside the 7×7 view whenever the agent faces it, and the agent did reach the goal six times under that view.

Alternative explanations this evidence rules out.

- *"Partial observability is the only problem."* Given the six successes, one could assume that with global observability the agent would complete the goal. The evidence above shows this is only partly true: the dominant deficit is reward sparsity, as credit fades quickly.
- *"The run is broken."* The same code reaches 100% on both 5×5 and 6×6.
- *"It only needs more steps."* Successes stop at step 202k and are never reinforced over the remaining ≈800k steps. More of the same budget would not help.

Temporary conclusion. In the problem statement of my proposal I claimed that the policy would collapse into high entropy that does not decay (1.94 at the start, 1.66 at the end vs. 0.02 at 6×6) and highly repetitive loops, causing the agent to stall out permanently. The evidence confirms this claim but contradicts my Metric 2, which predicted repetitive minimal loops, implying low entropy.

3 First comparison: control vs. interventions on 8×8

I run two interventions from the proposal against the control: three seeds each, the same 1M-step budget (976 × 1,024 steps), PPO code, hyperparameters (`conf/ppo/default.yaml`) and 7-action space. All runs are evaluated under the protocol on the same 200 layouts with no shaping (`eval.json` records `reward_shaping_at_eval: false`; G3 also records `fully_obs: true`).

Group (runs/cmp_…)	Success per seed s1 / s2 / s3	Mean ± sd [min–max]	Return / length	Stays > 0.9 from (steps)	Final entropy	Final expl. var.
G1 control `control_8x8_seed1–3`	0 / 0 / 100%	**33.3 ± 47.1%** [0.0–100.0]	0.321 / 435.5	— / — / 677k	1.7 / 1.8 / 0.2	−2.50 / −3.51 / 0.65
G2 shaped `shaped_8x8_seed1–3`	100 / 100 / 100%	**100.0 ± 0.0%** [100.0–100.0]	0.974 / 18.7	131k / 103k / 123k	0.02 / 0.01 / 0.02	0.95 / 0.95 / 0.96
G3 global view `fullobs_8x8_seed1–3`	7 / 5.5 / 3.5%	**5.3 ± 1.4%** [3.5–7.0]	0.024 / 626.6	— / — / —	1.9 / 1.8 / 1.8	0.51 / 0.21 / 0.62

Tab. 2 — Two interventions against the control.

Success = share of the 200 protocol episodes with return > 0 (`eval.json`). sd = population s.d. across seeds (`aggregate_eval.py`). Random policy: 1.5% on the same layouts. Per-seed columns are in seed order. The last three columns are from `metrics.csv` (entropy and explained variance: mean of the last 20 updates).

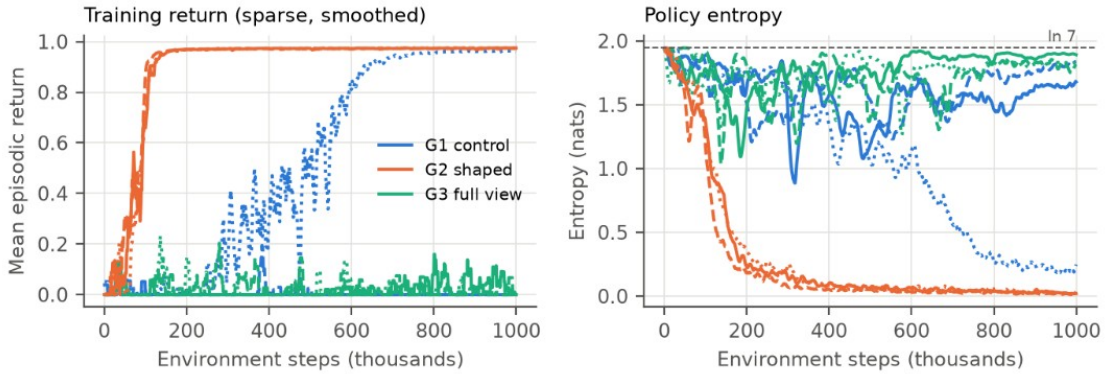


Fig. 6 — Training return (sparse for every group: statistics are logged before the bonus) and policy entropy for all nine runs. Colour = group; line style = seed (1 solid, 2 dashed, 3 dotted).

How to read these results under the rule fixed before the runs ("better" = the worst seed of one group beats the best seed of the other). **G2 vs. G1: not decisive.** Control seed 3 also reached 100%, so G2's worst seed only ties G1's best. Descriptively G2 is clearly ahead: 3/3 seeds solved vs. 1/3 (one-sided Fisher exact p = 0.20; three seeds cannot establish significance). G2 learns 5 to 6 times faster: its return stays above 0.9 from 103–131k steps, vs. 677k for the one successful control. Its episodes are also shorter (18.7 vs. 26.5 steps). **G3 vs. G1: not decisive either.** Full view gives 3.5–7.0% on every seed. That is above random and above the two failed controls, but no seed ever sustains a return above 0.5. In my proposal's case matrix, "G2 breaks out with a steep ascent while G3 stays near 0" corresponds to **Case 2: temporal credit deficit dominates**. With only three seeds I report this result as provisional.

Added value of the curves. With shaping, the curves show the critic becoming informative (explained variance 0.95) and entropy falling to 0.02 within ≈200k steps. Full view also makes the critic informative (0.21–0.62, vs. −2.5 / −3.5 in the failed controls) and produces 2 to 15 times more training successes (53–88 vs. 6–22). Yet entropy stays at 1.77–1.90: the policy still never commits. Control seed 3 escaped from the same start as the failed seeds (5–6 successes by 150k in all three) and then converged slowly. Early luck does not predict which seed escapes.

Validity checks. `cmp_control_8x8_seed1` reproduces `baseline_doorkey8x8_seed1` exactly (identical `metrics.csv` and `episodes.csv`; `compare_runs.py` differs only in the `exp_name` label). The shaped runs' episode logs contain only sparse returns (no bonus-valued entries). G1 and G2 ran at 55d359a, where the only untracked files were three figure PNGs. G3 ran at 5caf2f9 (clean tree), which adds only a rollout-buffer shape fix for full-grid inputs. The fix has no effect on 7×7 runs, as the exact seed-1 reproduction confirms.

4 Next step

With these results pointing to the temporal credit deficit, I continue with my original hypothesis, narrowed to that deficit. The supporting result is §3: milestone shaping solved 8×8 on all three seeds, 5 to 6 times faster than the only control seed that succeeded, while full observability did not solve it on any seed. Combined with §2 (key obtained, door never unlocked, uninformative critic), this points to credit assignment at the door step as the main cause, with vision second.

Two revisions follow from the data. (1) The 8×8 failure is probabilistic, not a fixed wall: one of three control seeds escaped. (2) Full view helps the critic but not the policy, so G3's remaining problem is committing to a behaviour, not missing information.

Plan for the final report. (a) At least 8 seeds per group, to turn the descriptive gap into a test and estimate the control's escape rate; (b) a shaping ablation (door-only vs. key-only bonus) to locate the bottleneck predicted in §2; (c) G2 at 16×16; (d) G2 + G3 combined, the Case 3 check. Metric 2's "repetitive local minima loops" will read "high-entropy repetitive loops", which is what §2 observed.

5 Protocol statement

Evaluation settings remain unchanged for every reported run: 200 episodes, evaluation seed 12345, the same layout seeds (episode *i* on 12345+*i*), sampled actions written to `eval.json`, success = return > 0. Group 2 is always evaluated without the milestone wrapper.

Network input. The network never sees a picture of the grid: MiniGrid provides a symbolic observation. Each observation is a dictionary with three parts, of which the agent uses two; the mission string is dropped. The image tensor (7×7×3: object id, colour id, state id) is one-hot encoded to 20 channels (11 + 6 + 3) and passed through three 3×3 convolutions. The facing direction is one-hot (4) and concatenated at the linear trunk.

Changes I made. Nothing in the PPO algorithm, the hyperparameters or the 7 discrete actions. (i) The encoder gains `AdaptiveAvgPool2d((1,1))` before the flatten. It adds no parameters and is the identity on the 7×7 view, so the baseline and G1/G2 are unchanged. (ii) G2 only: training reward + milestone bonuses. (iii) G3 only: the observation is the full N×N×3 grid (`FullyObsWrapper`), and the agent's own cell has its state channel set to 0 (MiniGrid writes the facing direction there, outside the 3 state classes; direction still arrives through the direction input). The 8×8 conv map (2×2) is averaged to the same 64 features. The rollout buffer takes the image shape from the environment instead of a hard-coded 7×7×3, which is unchanged for G1/G2. The evaluator rebuilds the G3 observation from the checkpoint.

AI use disclosure

I used Claude (Anthropic) to parse the run records and recorded episodes, compute the §1–§2 statistics and figures, review and fix the intervention code and tests, and draft this report. Every number is read from run artifacts. None was estimated. I used Gemini (Google) to rephrase and proofread the paper.

References

Chevalier-Boisvert, M. et al. (2023). Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *NeurIPS Datasets and Benchmarks.*

Huang, S. et al. (2022). CleanRL: High-quality single-file implementations of deep RL algorithms. *JMLR* 23(274).

Ng, A. Y., Harada, D., Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *ICML.*

Schulman, J. et al. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.