Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.38299

An Empirical Study of Task Progress Abstraction for Count-Based Exploration in MiniGrid DoorKey

Siyang Zhang

School of Electrical and Electronic Engineering, The University of Sheffield, Sheffield, UK
szhang218@sheffield.ac.uk

Abstract. Count-based exploration depends on which experiences share a visitation count. This study examines a hand-specified Task Progress Count Bonus (TPCB) that combines position with three current stages in MiniGrid DoorKey: a locked door without the key carried, a locked door with the key carried, and an unlocked door. Six conditions use identical symbolic policy inputs. Following development and random-trajectory bonus calibration, 60 confirmation runs cover two grid sizes and five new training seeds, with 100,096 interactions per run. Each model is evaluated on 100 reserved maps under stochastic and deterministic execution. TPCB reaches 100% mean stochastic success in DoorKey6. In DoorKey8, its 67.8% success compares with 66.2% for a compact-context control; the paired difference is 1.6 percentage points, with a 95% bootstrap interval of -31.2 to 42.6 points. A subsequent reward-schedule study uses 27 development runs and 15 fresh-seed validation runs. The selected TPCB setting reaches 96.0% during development but 49.4% in validation, versus 85.0% for the selected compact-context setting. This follow-up compares complete selected configurations, not count representations alone. TPCB's deterministic success in the original DoorKey8 confirmation is only 2.4%; a separate development diagnostic verifies cycles in all 179 deterministic failures examined. The results support task-stage counting in the smaller setting, but neither a consistent advantage over informative controls nor reliable deterministic execution. Privileged inputs, few training seeds and fixed map panels limit generalisation.

Keywords: Reinforcement learning, count-based exploration, task progress abstraction, MiniGrid DoorKey, policy evaluation

1. Introduction

In a sparse-reward task, useful behaviour may depend on actions that produce no immediate external reward. Count-based exploration encourages visits to infrequently encountered states, but the definition of a state for counting matters. Density-based pseudo-counts and hash-based counts extend this principle beyond small tabular spaces [1, 2]. A count representation still determines which experiences share a visitation history. A coarse representation may merge situations requiring different actions, while a detailed representation can distinguish changes that are unnecessary for completing the task.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38299

DoorKey in MiniGrid offers a small setting in which to examine this choice [3]. The same location can be visited before obtaining a key, while carrying it, or after unlocking a door. The candidate examined here, Task Progress Count Bonus (TPCB), combines location with three manually defined current stages. Task-specific exploration is already addressed by Learnable Episodic Count (LECO) and predictive-bisimulation exploration [4, 5]. The question is therefore not whether task awareness is new, but whether this particular simplification retains useful performance relative to informative counting controls.

This study makes three empirical contributions:

1) It compares task-stage counting with compact-context and direction-augmented controls under identical policy observations, testing whether a three-stage count key offers benefits beyond the inclusion of task information.

2) It evaluates fixed configurations after calibrating accumulated count bonuses on shared random-action trajectories. The analysis distinguishes this calibration from reward exposure during learning, which remains policy dependent. A separate follow-up selects reward schedules on development seeds and tests the selected configurations on fresh seeds and maps.

3) It separates exploratory development from new-seed confirmation and evaluates both stochastic and deterministic execution. Paired seed-level uncertainty and step-level cycle diagnostics distinguish sampled-policy performance from execution reliability, addressing established concerns about statistical evaluation in reinforcement learning [6].

The experiments address three questions: whether task-stage counting retains useful performance against informative controls, whether favourable results after reward-schedule selection persist on fresh training seeds, and whether stochastic success translates into reliable maximum-probability execution. The original confirmation resolves neither superiority nor equivalence in DoorKey8. The follow-up finds substantial variation after tuning, while the execution diagnostic identifies persistent action cycles. Together, these results define the conditions under which this particular count representation was tested and the limits of the favourable development results.

2. Related work

2.1. Counting and task-relevant novelty

Bellemare et al. derive pseudo-counts from density models, whereas Tang et al. count hashed state representations [1, 2]. TPCB uses an explicit symbolic key rather than either learned density estimation or a general-purpose hashing scheme. Its three-stage rule is a task-specific engineering choice, not a theoretical guarantee that the aggregated states share transition dynamics or decision value.

LECO combines learned discrete representations with episodic counting and task-specific reward modulation [4]. Their MiniGrid experiments describe task-irrelevant pickup and drop behaviour in a count-only variant. These observations rule out treating irrelevant-action repetition as a newly discovered general phenomenon. The diagnostic here instead verifies physical state-action cycles under maximum-probability execution in the saved development models. Liang et al. couple learned task-relevant representations and exploration through a predictive bisimulation metric [5]. Neither method is reimplemented in this study; the comparisons therefore concern symbolic count keys, not performance against these learned exploration systems.

DOI: 10.54254/2755-2721/2026.38299

2.2. MiniGrid experiments and evaluation scope

Kayal et al. compare intrinsic-reward mechanisms in MiniGrid and examine how results depend on observation format and exploration behaviour [7]. The present study addresses a narrower question within counting, holding the policy input fixed while varying the count key. Its privileged symbolic features and short training budgets prevent a direct numerical comparison with their experiments or an inference about RGB learning.

Henaff et al. distinguish global and episodic novelty and examine their dependence on structure shared across contextual tasks [8]. TPCB retains counts across episodes; consequently, identical keys may combine visits from different layouts. An episodic-reset comparison is outside the completed experiment. Finally, the uncertainty analysis follows the motivation for reporting variability under limited independent training runs [6], but uses a small, single-task paired bootstrap. It is not a reproduction of a multi-benchmark evaluation framework.

3. Method

3.1. Task and policy observations

The study considers the single-door DoorKey task in MiniGrid, a library of goal-oriented reinforcement learning environments [3]. An agent must obtain a matching key, unlock a door and reach a goal. DoorKey6 and DoorKey8 use grids of size 6 by 6 and 8 by 8, with episode limits of 360 and 640 steps, respectively. In the installed implementation, successful goal arrival yields one minus 0.9 times the fraction of the episode limit consumed. Other transitions receive zero external reward. Both environments retain the original seven actions, including pickup, drop, toggle and done. No action masking or automatic loop-breaking rule is applied.

All methods receive the same 156-dimensional vector: a normalised 7 by 7 symbolic observations with three channels, a four-dimensional direction indicator, two normalised absolute coordinates, and indicators for carrying a key and for the door being open or locked. The three image channels are divided by 10, 5 and 2, respectively; coordinates are divided by grid width minus one and grid height minus one. Absolute position and global door status are privileged simulator information. The experiment therefore concerns count representations under shared symbolic inputs, rather than learning task progress from pixels or exclusively local observations. The original comparison varies the count key with the calibrated coefficients described below; the candidate does not receive a larger policy observation. The follow-up additionally varies the coefficient schedule.

3.2. Task progress and counting representations

The candidate, denoted TPCB, counts the combination of position and current task stage. Stage zero means that the door is locked and the agent is not carrying its matching key. Stage one means that the door remains locked and the matching key is carried. Stage two means that the door has been unlocked. This definition is a function of the current simulator state, not a monotonic record of previous achievements. Dropping the key before unlocking returns the stage to zero. Closing an already unlocked door leaves the stage at two.

Table 1 lists the six conditions, including a Proximal Policy Optimization (PPO) baseline without an intrinsic bonus. The full-context count, labelled CACB in the preceding course project, includes direction, carried-object type and colour, and the door's location, colour, open state and lock state. Context Compact removes direction and door location but retains carried-object and door-status

DOI: 10.54254/2755-2721/2026.38299

details. It provides a closer information control for TPCB than the full-context method alone. TPCB Dir adds directions to the candidate's count key.

Table 1. Information used by each counting condition. Policy observations are identical across all six conditions

Condition	Count key
PPO	No count bonus
Position	Agent position
CACB	Position, direction, carried-object type and colour, full door descriptor
TPCB	Position and current task stage
TPCB Dir	Position, direction and current task stage
Context Compact	Position, carried-object type and colour, door open and locked flags

3.3. Bonus update and implementation

After each environment transition, the count of the reached key is incremented. The intrinsic bonus is then the method's coefficient beta divided by the square root of this updated count. The reward supplied to the learner is the sum of the external reward and this bonus. Initial states created by reset are not counted, whereas reached terminal states are counted. Count tables persist across episodes within a training run and are reset between independent runs. Key acquisition and door unlocking receive no separate event reward. All five count tables are recorded for diagnostics even when only one supplies the bonus; this implementation is therefore not an isolated runtime benchmark of count-key compression.

Persisting counts across procedurally generated episodes can combine visits from different layouts under one key. This is part of the tested design, not an assumption that all shared keys have identical decision value. Previous research distinguishes global and episodic novelty bonuses and relates their usefulness to structure shared across episodes [8]. The present experiment retains lifetime counts and does not compare against an episodic-reset variant.

The policy learner is PPO [9], implemented with Stable-Baselines3 [10]. The present study adds the current-stage abstraction, matched information controls, reward-scale checks, separated development and confirmation stages, and step-level execution diagnostics.

4. Experimental design

4.1. Development and confirmation

Development comprised 78 training runs using seeds zero, one and two. These runs examined training budget, information controls, reward coefficients and environment size. They informed a confirmation plan recorded before new training began. The confirmation stage used all six methods in both environments, with training seeds 10 through 14. This produced 60 separate training runs. All started from scratch and used the final checkpoint after the prescribed budget, without early stopping, best-checkpoint selection or extra training for individual failures.

Each run requested 100,000 interactions. With rollout length 128, the actual budget was 100,096 interactions, giving 6,005,760 training interactions in confirmation. Calibration trajectories, evaluations, unit tests and earlier team experiments are excluded from this total. Policy and value

DOI: 10.54254/2755-2721/2026.38299

networks each used two hidden layers of 64 units. The learning rate was 0.00025, minibatch size 64, optimisation epochs per rollout four, discount factor 0.99 and generalised advantage estimation parameter 0.95. PPO clipping was 0.2, the entropy coefficient 0, the value-loss coefficient 0.5 and the gradient-norm limit 0.5; advantages were normalised. Both networks used Tanh activations, orthogonal initialisation and Adam optimisation with the installed Stable-Baselines3 policy defaults. Value-function clipping and target-KL early stopping were disabled. Each process used one central processing unit (CPU) computation thread; at most two runs executed concurrently. Software versions were Python 3.12.14, PyTorch 2.13.0+cpu, Gymnasium 1.3.0, MiniGrid 3.1.0 and Stable-Baselines3 2.9.0.

4.2. Reward coefficient control

The coefficient for each count representation was fixed before confirmation. At a reference coefficient of 0.01, each representation accumulated a total bonus on a prescribed uniform-random action trajectory. Its calibrated coefficient was 0.01 multiplied by the ratio of the CACB total to its own total. DoorKey6 used one trajectory of 50,048 interactions. DoorKey8 used five such trajectories and the ratio of their pooled totals. All count representations were measured on the same trajectory within each calibration run. This procedure equates pooled bonus totals on the fitting trajectories, not on trajectories later generated by learned policies.

Table 2 gives the frozen coefficients. DoorKey8 used its environment-specific recalibration even though some development means had been higher with transferred DoorKey6 coefficients. No confirmation scores were used to revise this decision. The difference between one and five calibration trajectories reflects the historical development procedure. Consequently, confirmation assesses the repeatability of these fixed configurations; it does not establish equal-quality optimal tuning for every method or an identical calibration sample budget across environments.

Table 2. Frozen bonus coefficients, displayed to eight significant figures. PPO has no intrinsic bonus. Training used the full-precision calibrated coefficients; the displayed values are rounded

Condition	DoorKey6 beta	DoorKey8 beta
PPO	Inactive	Inactive
Position	0.062996583	0.069768162
CACB	0.010000000	0.010000000
TPCB	0.042001502	0.050530922
TPCB Dir	0.021608685	0.026484654
Context Compact	0.034269543	0.047444047

4.3. Evaluation and uncertainty

For training seed s, map generation used 100,000 times s plus the episode index. Methods with the same training seed therefore shared the map-seed schedule by episode number, but their different episode lengths could produce different numbers of encountered maps. Distinct generation seeds also do not guarantee structurally distinct layouts in these finite environments.

Development learning curves used 20 fixed maps, with generation seeds starting at 10,000,000, at initialization and after completed PPO updates crossing each 10,000-interaction interval. Final

DOI: 10.54254/2755-2721/2026.38299

development evaluations used 100 maps starting at 11,000,000. The action-sampling seeds for development curves and final development evaluation were 12,000,000 and 13,000,000, respectively. These measurements were retained as process records and were not used for checkpoint selection in confirmation. All 60 final models and training files were checked and locked with cryptographic hashes before formal evaluation used the reserved map seeds 20,000,000 through 20,000,099.

Every model was evaluated on these 100 reserved maps in both stochastic and deterministic modes, using only external rewards. Stochastic evaluation sampled from the policy distribution; deterministic evaluation selected its maximum-probability action. The action seed was 21,000,000 and inference batch size was 20. Python, NumPy and PyTorch random states were isolated during evaluation. Reported successes are first calculated per model and then summarised across the five training seeds. Evaluation episodes are not treated as independent training replications.

The prespecified primary comparison was TPCB minus Context Compact in DoorKey8 stochastic success rate. Its interval was computed by resampling the five paired training-seed differences with replacement 50,000 times and taking the 2.5th and 97.5th percentiles, with analysis seed 22,000,000. Other comparisons are descriptive. Here, five-seed bootstrap intervals may be unstable and describe training-seed variation conditional on a fixed map panel and action-sampling protocol. They do not quantify all uncertainty over future maps or training configurations.

All 12,000 formal evaluation episodes passed raw-data and summary checks. Twelve models, selected in advance as training seed 10 for each method and environment, were reloaded and evaluated again in both modes. All 2,400 replayed episodes matched their saved records. The implementation and locked training-file hashes were unchanged.

4.4. Follow-up reward schedule validation

After the original confirmation, a separate DoorKey8 study examined whether reward-schedule selection yielded more reliable performance. TPCB, Context Compact and CACB each used three candidates: the original constant coefficient, a constant coefficient at half that value, and linear decay from the original value at the first interaction to zero at interaction 100,096. Development used training seeds 30-32, giving 27 runs. Each run used 100,096 interactions, the same policy observations and PPO configuration, and its final checkpoint. For each method, the candidate with the highest mean final stochastic success across those three seeds was selected.

The selected settings were constant half-strength for TPCB and CACB, and linear decay for Context Compact. Half-strength keeps the coefficient at one half of its base value. Linear decay reduces its multiplier evenly from one at transition 1 to zero at transition 100,096; both schedules therefore have the same sum of coefficient multipliers, but not necessarily the same realised intrinsic reward. These settings were locked before training seeds 40-44 produced 15 new models, again with 100,096 interactions per run. All models were sealed before evaluation on a new panel of 100 maps in both execution modes, yielding 3,000 validation episodes. Base coefficients were the DoorKey8 values in Table 2. Final schedule-development evaluation used map seeds 31,000,000-31,000,099 and action seed 33,000,000. Validation used map seeds 36,000,000-36,000,099, action seed 37,000,000 and inference batches of 20. In constant, linear-decay and constant-half order, development means were 65.7%, 83.7% and 96.0% for TPCB; 83.0%, 94.0% and 65.7% for Context Compact; and 64.7%, 63.0% and 67.7% for CACB. No validation score was used to select a schedule or checkpoint.

This follow-up was not part of the original confirmation plan and its results are reported separately. It compares three selected method-and-schedule configurations, rather than isolating a

DOI: 10.54254/2755-2721/2026.38299

causal effect of the count key. Each method received the same three-candidate search in this follow-up, but earlier development budgets were not identical. The main follow-up contrast is selected TPCB minus selected Context Compact in stochastic success, with a paired training-seed bootstrap using 50,000 resamples and analysis seed 39,000,000. As in the original analysis, five seeds provide limited precision and the interval is conditional on the fixed evaluation panel.

Across the original study and this follow-up, 180 completed training runs used 16,515,840 interactions. Two technically interrupted validation attempts had at least 70,144 additional recorded interactions and were excluded from the completed-run total, not treated as independent replications. Calibration, evaluation and earlier team runs are outside this total. No completed failure seed was excluded.

5. Results

5.1. Prespecified comparison in DoorKey8

Figure 1 summarises the prespecified comparison. TPCB achieved mean stochastic success of 67.8% in DoorKey8, compared with 66.2% for Context Compact. The mean paired difference was 1.6 percentage points, with a 95% bootstrap interval from -31.2 to 42.6 points. This result does not provide clear evidence that TPCB improves on the strong compact-context control. The wide interval also prevents an equivalence conclusion: lack of demonstrated superiority is not evidence that the methods perform identically.

The five TPCB success rates were 80%, 11%, 98%, 98% and 52%, in training-seed order. Context Compact obtained 0%, 39%, 97%, 99% and 96%. High success rates in some candidate runs therefore coexist with substantial failures, and the relative ordering changes across seeds. CACB and TPCB Dir had higher mean stochastic success than TPCB in this environment, at 70.6% and 73.8%, respectively.

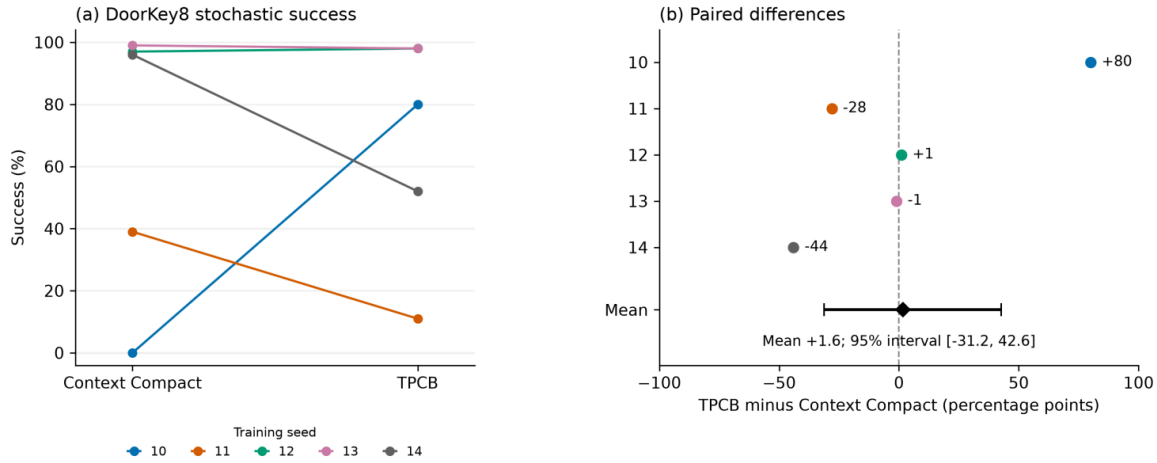


Figure 1. Prespecified DoorKey8 comparison. Lines pair training seeds 10-14; each rate uses 100 reserved maps. The diamond and interval show the mean paired difference and the 95% percentile bootstrap interval over five seeds. Uncertainty is conditional on the fixed evaluation panel; five-seed intervals may be unstable (picture credit: original)

DOI: 10.54254/2755-2721/2026.38299

5.2. Environment size and learning curves

In DoorKey6, TPCB and CACB both succeeded in all recorded stochastic test episodes, while Context Compact averaged 99.4%. These near-ceiling results support the narrower observation that task-stage counting can retain high final performance in this setting. They provide little resolution for claiming an advantage over the stronger controls. Zero observed seed standard deviation at 100% is not evidence of zero uncertainty outside this finite sample. Position counting remained variable, with individual success rates of 1%, 98%, 1%, 0% and 100%.

Final success and learning-curve performance did not give identical rankings. In the confirmation training runs, mean normalised development-curve area in DoorKey8 was 0.4453 for TPCB and 0.5098 for Context Compact. The corresponding DoorKey6 values were 0.6198 and 0.6048. Normalised curve area is the trapezoidal integral of success against actual interaction steps divided by the final interaction count. These curves use the development map panel and sparse 20-episode assessments, not the reserved formal panel. They should therefore be interpreted as descriptive process measurements, with no claim that TPCB consistently improves sample efficiency.

5.3. Deterministic execution and diagnostic evidence

Figure 2 displays all method-by-seed results from the original confirmation. The evaluation mode substantially changed the observed performance. DoorKey6 mean deterministic success was 80.0% for TPCB, 60.8% for Context Compact and 44.2% for CACB. In DoorKey8, these means were only 2.4%, 4.8% and 0.0%. Stochastic success therefore did not imply reliable execution of the policy's most probable action.

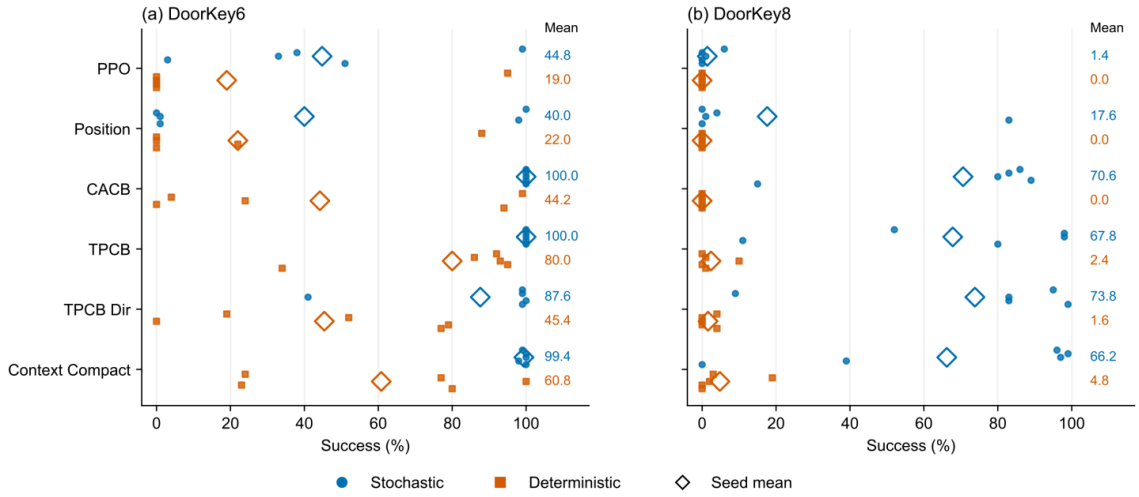


Figure 2. Formal stochastic and deterministic success. Small points show all five training seeds per condition; diamonds and printed values show seed means. Minor vertical offsets separate points, without changing success values. Each model uses 100 reserved maps per mode (picture credit: original)

Before confirmation, a separate diagnostic examined nine development models: TPCB, Context Compact and CACB at training seeds zero, one and two. Each model was run on the first 20 maps of the existing development panel in both execution modes. These 360 episodes are not independent confirmation evidence. Step records included action probabilities, observations, position, direction, carried objects and full-grid state fingerprints. A repeated coordinate alone was insufficient to

DOI: 10.54254/2755-2721/2026.38299

classify a loop; the subsequent physical states, observations and actions had to follow the same cycle through the remaining rollout.

Figure 3 illustrates one verified cycle. All 179 deterministic failures in this diagnostic had verified cycles: 67 repeated ineffective forward moves, 42 ineffective toggles, 40 ineffective pickups, 25 rotation cycles and five other action cycles. For example, a TPCB development model facing a key assigned probability 0.801 to moving forward and 0.105 to pickup. Repeated maximum-probability selection left its physical state unchanged, whereas stochastic execution on that map reached the goal in 61 steps. This identifies an immediate behavioural explanation for failure in the diagnosed cases. It does not establish which training component caused those action probabilities, nor does it imply that every formal-test failure has the same verified mechanism.

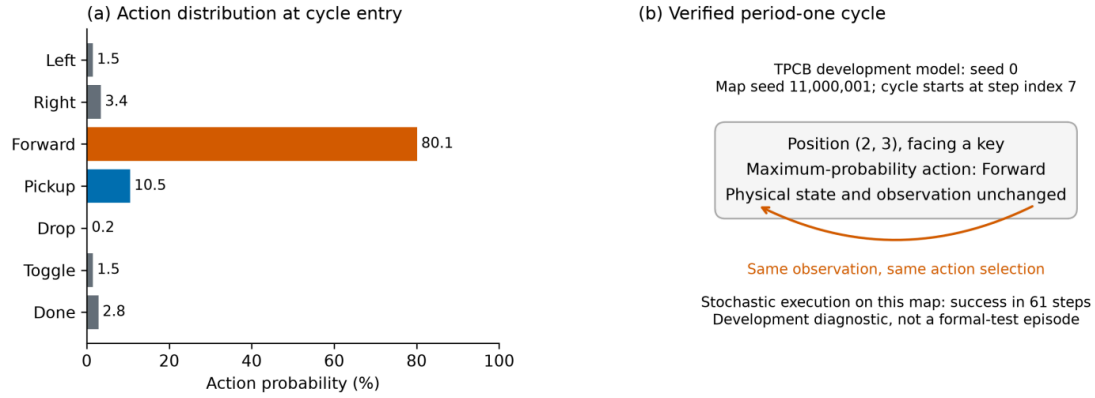


Figure 3. Development-model execution diagnostic. The action distribution and period-one cycle come from the saved TPCB seed-0 case on map 11,000,001. Stochastic execution succeeds in 61 steps. The cycle explains the observed deterministic behaviour, not its training cause (picture credit: original)

5.4. Performance after reward schedule selection

Table 3 compares the selected development settings with their fresh-seed validation. TPCB averaged 96.0% stochastic success on its three development seeds but 49.4% on the five validation seeds. Context Compact and CACB reached validation means of 85.0% and 69.4%, respectively. TPCB validation scores were 82%, 0%, 76%, 0% and 89% in seed order. The low mean therefore reflects large between-run variation, including two runs with no successful test episodes, rather than uniformly moderate performance.

Table 3. DoorKey8 stochastic success after schedule selection. Development means use three training seeds; validation values are mean and sample standard deviation over five new training seeds. All values are percentages. Different map and action-seed panels were used in the two phases, so the phase difference is descriptive, not a paired causal estimate

Selected configuration	Development mean	Validation mean (SD)
TPCB, constant half	96.0	49.4 (45.3)
Context Compact, linear decay	94.0	85.0 (19.4)
CACB, constant half	67.7	69.4 (33.2)

DOI: 10.54254/2755-2721/2026.38299

The selected TPCB-minus-Context-Compact difference was -35.6 percentage points, with a paired 95% bootstrap interval from -77.2 to 6.0 points. The estimate is unfavourable to TPCB but does not establish statistical inferiority. Nor does the interval establish equivalence. Deterministic success remained low: 1.4% for TPCB, 0.8% for Context Compact and 0.0% for CACB. The tested schedule changes did not resolve the execution-mode gap.

The development scores were used for schedule selection and are therefore not independent estimates of the selected configurations' performance. Fresh training seeds, maps and action-sampling seeds all changed at validation, so the drop from 96.0% to 49.4% cannot be attributed solely to overfitting, schedule choice or map generalisation. The direct conclusion is narrower: the high development mean did not recur in this validation batch. These results do not identify a universally best schedule and are not pooled with the original confirmation.

6. Discussion and limitations

6.1. Interpretation of the counting abstraction

The compact-context comparison changes the interpretation of the candidate. Performance against plain PPO or position counting alone would suggest a large benefit from adding task information, but would not establish that a three-stage description is preferable to other informative keys. The DoorKey8 confirmation does not resolve that finer comparison. TPCB simplifies the count key, not the policy observation, so this experiment does not demonstrate that a compressed state is sufficient for control.

Reward calibration also remains conditional. Equal bonus totals on random-action traces do not imply equal bonuses after the policies visit different states. A representation's effect on reward exposure is therefore not fully separated from its effect on visitation. Neither the present calibration nor the fixed training budget establishes optimal tuning. The schedule follow-up tests one practical response to this limitation, but its selected configurations still combine representation and coefficient choices. Its fresh-seed results do not support presenting the best development score as a dependable performance gain.

6.2. Execution and limits of interpretation

For the diagnosed feedforward policies, revisiting an identical observation under maximum-probability selection reproduces the action choice. The verified cycles explain immediate execution failure; they do not identify why training produced those probabilities. Sampling can change the trajectory but is neither a newly introduced remedy nor a guarantee of success. Further interventions on training or action masking would require their own experiments.

The original confirmation covers two sizes of one task family; the follow-up covers only DoorKey8. Both use privileged symbolic features, five evaluation-stage training seeds per configuration and a fixed map panel within each stage. They do not establish performance in visually learned or multi-door tasks. Further research could isolate episodic versus persistent counts or intervene on training factors behind the execution gap. Such changes would require new development and fresh confirmation data, rather than repeated selection on either panel already examined.

DOI: 10.54254/2755-2721/2026.38299

7. Conclusion

Task progress was examined as a manually specified representation for count-based exploration, using the same policy observations across six conditions. The experiment separated exploratory development from a fixed new-seed confirmation and retained both sampled and maximum-probability policy execution. This design makes a comparison about a defined count key under stated conditions, rather than about an unspecified improvement to reinforcement learning.

The principal result is conditional. TPCB retained high stochastic performance in DoorKey6, but its slightly higher mean than the compact-context control in DoorKey8 was accompanied by substantial seed variation. The uncertainty interval supports neither a superiority claim nor an equivalence claim. The detailed-context and direction-augmented controls also remain important: the candidate was not the highest-mean method in the larger environment. These findings limit the interpretation of favourable development runs. The subsequent schedule study strengthens that caution: the selected TPCB setting achieved a high development mean, yet it did not retain that level across fresh validation seeds and maps. This follow-up is evidence about the tested selection procedure, not proof that the stage representation is intrinsically inferior.

Execution diagnostics provide a second result. Success obtained by sampling actions did not translate into reliable maximum-probability execution. Repeated physical state-action cycles accounted for the failures examined in the development diagnostic, while their training causes remain unresolved. Those observations should not be generalised to unexamined failures or presented as a complete fix.

The main limitations are the narrow task family, privileged symbolic inputs and limited seed replication. The training causes of deterministic cycles remain unresolved. Further study should isolate a prespecified intervention, such as episodic rather than persistent counting, and use fresh training seeds and evaluation panels. The present evidence supports task-stage counting in a restricted setting but does not establish a general performance advantage.

References

[1] Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems, 29*, 1471–1479.

[2] Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2017). #Exploration: A study of count-based exploration for deep reinforcement learning. *Advances in Neural Information Processing Systems, 30*, 2753–2762.

[3] Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & KTerry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems, 36*.

[4] Jo, D., Kim, S., Nam, D. W., Kwon, T., Rho, S., Kim, J., & Lee, D. (2022). LECO: Learnable episodic count for task-specific intrinsic reward. *Advances in Neural Information Processing Systems, 35*, 30432–30445.

[5] Liang, D., Liu, R., Wan, L., Liu, Y., & An, B. (2026). Task-aware exploration via a predictive bisimulation metric. *arXiv preprint arXiv:2602.18724*.

[6] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., & Bellemare, M. G. (2021). Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems, 34*, 29304–29320.

[7] Kayal, A., Pignatelli, E., & Toni, L. (2025). The impact of intrinsic rewards on exploration in reinforcement learning. *Neural Computing and Applications, 37*, 16269–16303.

[8] Henaff, M., Jiang, M., & Raileanu, R. (2023). A study of global and episodic bonuses for exploration in contextual MDPs. In *Proceedings of the 40th International Conference on Machine Learning (Vol. 202)*, pp. 12972–12999. PMLR.

DOI: 10.54254/2755-2721/2026.38299

[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
[10] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research, 22*(268), 1–8.