BEYOND TOKEN-AVERAGE LOSS: REBALANCING CONTROL AND BINDING IN AGENT POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Distilling multi-step agent policies into autoregressive language models requires mastering two coupled decisions: high-level operational control and granular argument binding. Under standard action serialization, control opcodes span only one or two tokens, whereas arguments, identifiers, and syntax dominate sequence length. Consequently, standard token-mean cross-entropy subordinates critical control decisions to argument formatting, leading to brittle failures where a single mispredicted opcode derails an execution trajectory. To address this structural imbalance, our action-group-normalized objective (L_{BAL}) equalizes supervisory weight between control opcodes and argument bindings at each step. In a controlled 2×2 factorial study on a shared parameter-binding-adapted Qwen3-4B backbone, we evaluate standard cross-entropy (L_{CE}) versus balanced L_{BAL} across expert demonstrations and matched replay substitution, strictly matching sample exposure and update budgets across three seeds. During evaluation, student policies operate fully autonomously, generating opcodes and arguments end-to-end without external guidance, policy masks, or runtime heuristics. The balanced L_{BAL} objective yields double-digit improvements in closed-loop task success: +12.22 percentage points on replay data (95% task-cluster bootstrap interval: [9.17, 15.42], conditional on observed seeds) and +10.56 points on expert data. Fully autonomous balanced students achieve 94.17% (replay) and 95.00% (expert) mean success, matching the 95.00% point estimate of an external teacher-guided reference. In contrast, expanding offline recovery-state coverage yields no benefit over expert demonstrations under standard objectives, demonstrating that balanced supervisory allocation governs policy competence where offline state diversity alone provides no benefit. Finally, stress tests under joint tool-and-control shifts map the operational boundaries of frozen policies. Code and recorded outcomes are provided in the supplementary materials.

1 INTRODUCTION

Autonomous agent policies must execute two intertwined decisions at every step: selecting which high-level operation to perform (*control*) and binding that operation to valid entities, arguments, and structured syntax (*binding*). Selecting `WAIT` is futile if the target handle specifies the wrong workers; conversely, flawless argument generation cannot advance a task if the policy repeatedly verifies rather than synchronizing or terminating. When distilling these capabilities into autoregressive language models, full-action serialization introduces a structural asymmetry: operational control codes (opcodes) typically span only one or two tokens, whereas entity references, argument payloads, and syntactic delimiters dominate the sequence. Consequently, standard token-mean cross-entropy subordinates critical control to argument formatting purely by sequence length.

This work investigates whether rebalancing this loss allocation directly improves autonomous closed-loop execution, and isolates this effect from the coverage of training states. We establish a controlled 2×2 factorial experiment crossing two loss formulations, token-average cross-entropy (L_{CE}) and equal group-normalized loss (L_{BAL}), with two equal-sized training distributions: expert trajectories and a matched replay-replacement table. All four experimental arms evaluate fine-tuning from a shared parameter-binding-adapted Qwen3-4B parent across three random seeds. Within each

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

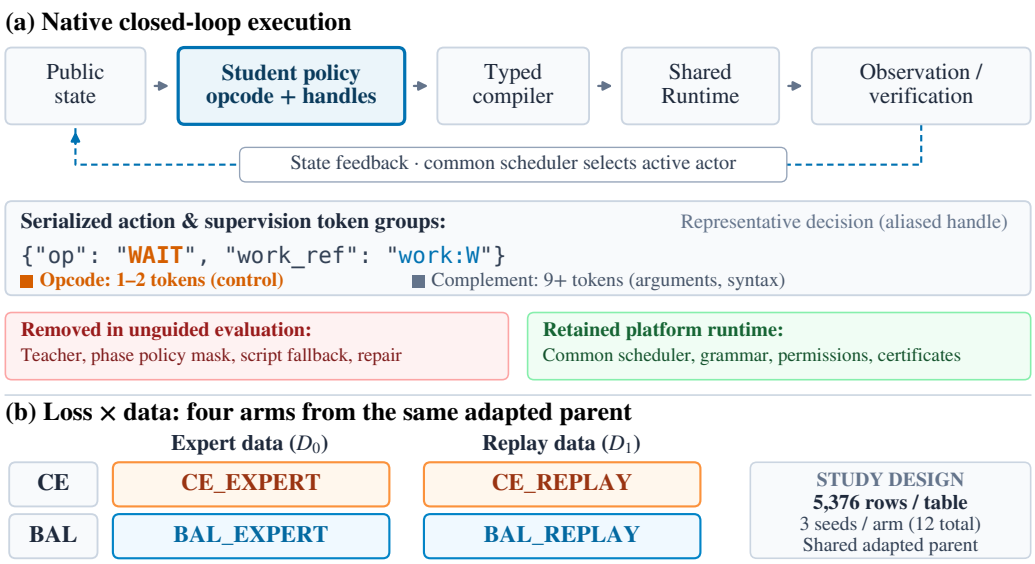


Figure 1: Overview of the experimental framework. (a) Closed-loop policy execution: the student generates control opcodes and argument bindings without external opcode guidance or policy masks, operating within a shared platform runtime. (b) The 2×2 factorial design crosses loss formulations (L_{CE} vs. L_{BAL}) with training distributions (D_0 vs. D_1) across three random seeds.

data distribution, the comparison strictly preserves sample instances, sequence order, and optimizer update budgets.

To evaluate genuine autonomous policy execution, student inference removes all external scaffolding: students receive no opcode teacher, phase-specific policy masks, script fallbacks, or semantic argument repairs. Operating within a shared platform runtime that enforces universal actor scheduling, typed permissions, and certificate verification, this protocol directly measures whether the distilled policy has acquired reliable, self-directed control and binding.

Our empirical findings demonstrate that equal group weighting delivers substantial gains across closed-loop environments: L_{BAL} improves mean task success by 12.22 percentage points on replay data (95% task-cluster bootstrap interval: [9.17, 15.42]) and by 10.56 points on expert data, elevating autonomous student success rates to 94.17% and 95.00% (approaching the 95.00% benchmark of an external teacher-guided reference). In contrast, expanding offline recovery-state coverage yields no detectable benefit over expert demonstrations under standard token-mean objectives, demonstrating that supervisory allocation governs policy competence here rather than passive state coverage. Detailed error diagnostics reveal that group normalization specifically rescues control failure modes, such as premature synchronization during multi-branch tasks, while stress testing under joint tool-semantic and aggregation shifts maps the operational boundaries of frozen policies.

We make three primary contributions: first, an explicit formulation of the opcode/complement token-length asymmetry and a group-normalized objective (L_{BAL}) evaluated under an unguided, autonomous execution protocol; second, a controlled 2×2 closed-loop comparison demonstrating that balancing control supervision yields double-digit success improvements (+12.22 pp and +10.56 pp), whereas offline recovery-state expansion alone cannot compensate for diluted control supervision; and third, diagnostic characterization of execution failure modes across concurrency levels, resource states, and training seeds, mapping concrete operational boundaries. Figure 1 illustrates the framework.

2 Related Work and Scope

Policy distillation transfers complex behaviors from teachers into compact, deployable student networks (Rusu et al., 2016). While early formulations focused on discrete actions in simulated games,

agent policy distillation in language models requires generating composite actions that combine syntactic boilerplate, operational opcodes, and concrete entity bindings.

Recent work applies action-aware token reweighting to specific language agent settings. Octopus v2 explores functional-token weighting for on-device function calling (Chen & Li, 2024), observing accelerated training convergence. FunReason balances reasoning steps against function-call supervision to improve tool-use fidelity (Hao et al., 2025), and GUI-Libra introduces action-aware token reweighting under partial execution verification for native graphical user interface (GUI) agents (Yang et al., 2026). In contrast, our work addresses a general structural imbalance in serialized agent supervision: decoupling the single control opcode from the remaining action complement within fully serialized execution, evaluated through closed-loop benchmarks and matched-table factorial controls.

Expanding learner-state coverage addresses the covariate shift inherent to sequential decision-making, where downstream states depend on the learner's own prior actions (Ross et al., 2011). In our controlled regime, we evaluate the specific efficacy of offline recovery-state replacement within an equal training data budget. This design isolates whether recovery-state coverage or control-loss allocation drives closed-loop task completion (Appendix Table A6).

3 STRUCTURED EXECUTION SETTING

Actions and observations. Let s denote the public state, o an opcode, and a its arguments. The autoregressive action distribution factors as:

$$\pi_\theta(o, a \mid s) = \pi_\theta(o \mid s)\, \pi_\theta(a \mid s, o). \tag{1}$$

The public state encapsulates active task obligations, available resources, actor ownership, runtime observations, and opaque candidate handles. Under a generic grammar, the model jointly generates the control opcode and its associated argument references. The compiler materializes the selected public values without semantic autofill or correction, and the Runtime executes the resulting typed action subject to shared actor permissions and capacity. Verified closed-loop execution progress and terminal state validation, rather than syntactic similarity to gold sequences, define task success.

Residual external mechanisms. A shared actor scheduler governs execution dispatch across distributed entities: root actions allocate and assign work, synchronize workers, integrate partial state, verify results, and terminate; worker actions invoke tools, record intermediate outputs, and return. To rigorously assess autonomous policy competence, native student evaluation completely removes external opcode teachers, phase-specific policy masks, and fallback heuristics, while executing within the host platform's standard actor permissions, typed compilation, and certificate-validation protocols.

Work-set representation. To structure multi-actor coordination, the public interface exposes every non-empty subset of the non-root work domain as candidate targets for synchronization (`WAIT`). The interface does not filter candidates to the correct owner set, synthesize missing parameters, or modify runtime semantics; the policy must autonomously identify and select the valid work-set membership. Because this interface representation is established identically across all conditions, observed performance differences reflect the models' learned decision policies rather than representational discrepancies.

In the representative execution shown in Figure 1, the root observes four returned workers and selects `WAIT` with candidate handle `work:W`. The compiler materializes precisely those four worker identities, and the recorded join succeeds. All alternative non-empty candidate subsets remain simultaneously selectable, confirming that the student selects the appropriate handle autonomously. Appendix C provides full observation excerpts, scheduler rules, and termination criteria.

Task scope. Our empirical evaluation uses a procedural partition-sum task family: worker agents execute scalar arithmetic over assigned integer subsets, and the root agent coordinates and integrates branch outputs into a verified result dictionary. Training tasks instantiate two and three branches ($n = 2, 3$), while evaluation introduces four branches ($n = 4$) to examine execution resilience under increased concurrent coordination.

Table 1: Structure and composition of the matched training tables (D_0 and D_1). Both tables contain exactly 5,376 rows spanning the same 192 semantic tasks. Recovery states are incorporated into D_1 via controlled replacement while strictly fixing total update volume.

Dataset Component / Coverage Condition	**Count or Status**
Total rows per table (D_0 and D_1, 192 tasks)	5,376 rows
Retained canonical expert rows in D_1	3,840 rows
Matched replacement rows in D_1 (learner rollouts)	1,536 rows
Novel control states in D_1 (absent from D_0)	326 states
Substituted recovery rows from failed learner episodes	237 rows
Noncanonical allocation / at-capacity valid reuse	Not covered

4 TRAINING INTERVENTIONS

4.1 MATCHED DATA TABLES

Both training tables contain 5,376 rows derived from the same 192 semantic training tasks. The expert table D_0 contains canonical trajectories supervised by the public program expert. The matched replay-replacement table D_1 preserves 3,840 expert rows while substituting 1,536 positions with candidate states drawn from a frozen learner rollout pass. Replacements are strictly matched by opcode, branch count, and initial-resource strata, utilizing deterministic token-length neighbor matching against the displaced expert instances. Length distributions remain tightly controlled; only 196 replacements differ from their target by more than a single 512-token bucket.

The substituted replay rows introduce 326 novel control states absent from D_0, including 237 states captured from previously failed learner episodes. Full-action supervision labels are provided by the public program expert with runtime execution validation. This intervention injects offline recovery diversity, such as managing idle workers following task completion, while maintaining equal data volume for direct factorial comparison (Table 1).

4.2 GROUP-NORMALIZED OBJECTIVE

For example i, let O_i denote the set of supervised tokens corresponding to the unquoted opcode value, and let R_i represent the complementary set of supervised action tokens (encompassing arguments, references, JSON syntax, and end-of-sequence (EOS) tokens). Boundary tokens overlapping multiple spans receive opcode priority. Denoting $\bar{\ell}_{O_i}$ and $\bar{\ell}_{R_i}$ as the respective group mean cross-entropies, the standard token-average objective and our balanced group-normalized objective are formulated as:

$$L_{\mathrm{CE}}^{(i)} = \frac{|O_i|\bar{\ell}_{O_i} + |R_i|\bar{\ell}_{R_i}}{|O_i| + |R_i|}, \tag{2}$$

$$L_{\mathrm{BAL}}^{(i)} = \tfrac{1}{2}\bar{\ell}_{O_i} + \tfrac{1}{2}\bar{\ell}_{R_i}. \tag{3}$$

Prompt and padding tokens are excluded, and reduction computes the mean over examples across the gradient-accumulation window. The method introduces no logit grammar masks or token duplication; L_{BAL} operates directly at the loss level, rebalancing intra-example token groups without skewing cross-corpus class margins.

Under this formulation, the effective objective coefficient assigned to an individual opcode token increases from $1/(|O_i| + |R_i|)$ under L_{CE} to $1/(2|O_i|)$ under L_{BAL}. Across the empirical training rows, the mean objective coefficient on the opcode-group mean loss is 3.35822% in D_0 and 3.35805% in D_1 (ranging from 2.63158% to 6.25% across individual rows), whereas L_{BAL} fixes this coefficient at an exact 50%. These values represent objective loss coefficients rather than measured empirical shares of realized gradients. Figure 2 shows the empirical distribution of these coefficients across all 5,376 training rows. Because opcode tokens are substantially outnumbered by arguments ($|O_i| \ll |R_i|$) and the sequence ratio $q_i = |O_i|/(|O_i| + |R_i|)$ fluctuates across examples, no global scalar multiplier applied to L_{CE} can establish consistent intra-example balance.

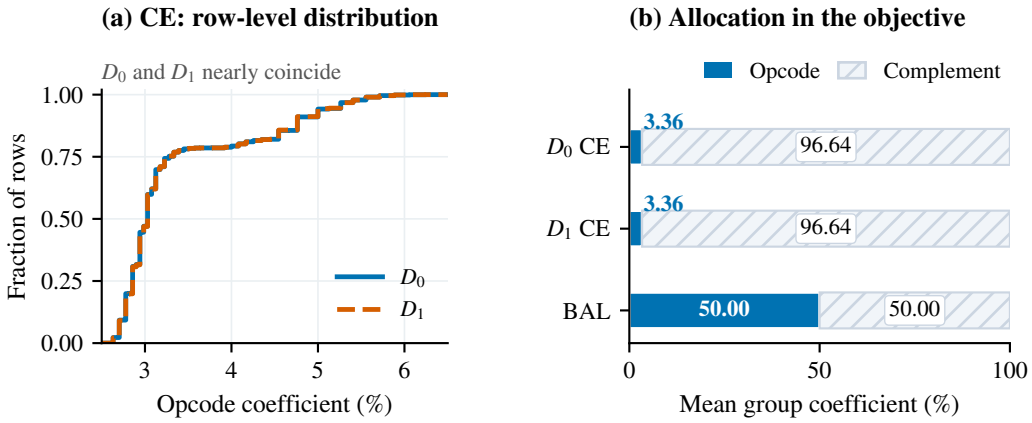


Figure 2: Distribution of opcode loss coefficients under token-average and group-normalized objectives. (a) Empirical cumulative distribution functions (ECDFs) of the opcode objective coefficient under L_{CE} across all 5,376 rows in D_0 and D_1 (mean $\approx 3.36\%$, range: 2.63–6.25%). (b) Comparison of nominal objective coefficient between opcode and complementary action tokens under L_{CE} versus group-balanced L_{BAL} (exact 50% allocation).

4.3 Optimization controls

All models originate from the open-source Qwen3-4B base model (Yang et al., 2025), initialized from a shared parameter-binding parent checkpoint developed on the declared 192 training tasks (lineage detailed in Appendix B). This ensures that all four factorial arms (CE_EXPERT, BAL_EXPERT, CE_REPLAY, and BAL_REPLAY) evaluate policy control learning from an identical parameter-binding foundation.

Each experimental arm trains for one complete epoch across three independent seeds (13, 37, 71), comprising 5,376 microsteps and 336 AdamW updates (totaling 4,032 updates across all 12 formal runs). Fine-tuning employs NF4 double quantization (Dettmers et al., 2023), BF16 mixed-precision computation, and LoRA adaptation (Hu et al., 2021) across all attention and MLP projection weights (full hyperparameters provided in Appendix A). Within each data table, this design strictly fixes instance exposure, example ordering, and total optimization steps.

5 Evaluation and estimands

Paired task benchmark. Evaluation is conducted on 120 held-out semantic base problems, balanced across branch counts $n \in \{2, 3, 4\}$ (40 bases per stratum). Each base problem is evaluated under two distinct resource environments: a clean root configuration and a configuration featuring one initially idle compatible worker. This creates 240 paired evaluation episodes per condition, totaling 3,600 recorded main episodes across the 12 trained students and three reference baselines. Additional fixed-state diagnostics include 60 counterfactual decision pairs and 72 canonical binding states evaluated across dual views.

Reference baselines. We examine three reference policies to contextualize student performance: `BIND_NATIVE` tests the unguided parameter-binding parent without opcode guidance; `BIND_TEACHER` pairs ground-truth opcodes with the parent model's argument generation; and `SCRIPT_INDEX` executes a deterministic step-index state machine. Autonomous students operate without external opcode supervision, policy masks, script interventions, or semantic parameter repairs.

Closed-loop success criteria. Episodes execute within the asynchronous platform Runtime using greedy decoding (seed 101), bounded by 24 decisions, 8 tool calls, 8,192 generated tokens, and 64 logical-time units. Standard syntax rules and permissions apply across all experimental conditions. Task completion strictly requires reaching a verified terminal state: producing the exact target output,

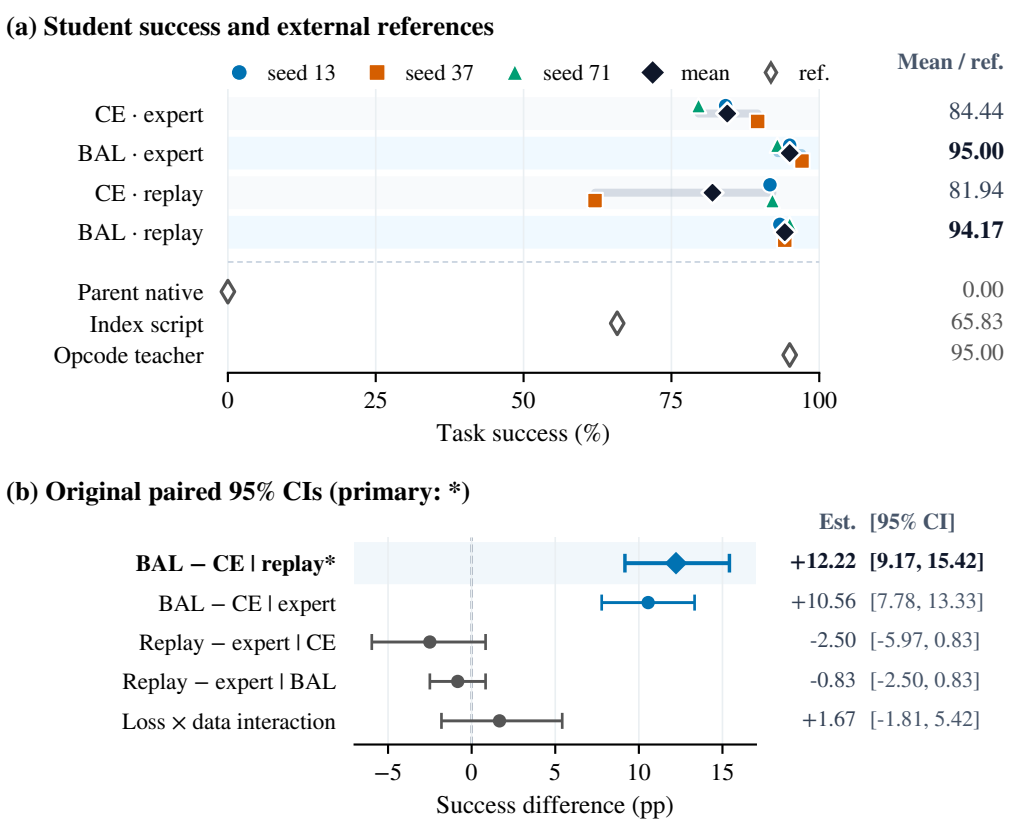


Figure 3: Closed-loop task success and factorial contrasts across evaluation environments. (a) Success rates across 120 held-out base tasks (240 episodes per model condition). Shapes denote individual random seeds; solid diamonds denote condition means, and hollow diamonds denote external references. (b) Factorial contrasts with 95% task-cluster bootstrap confidence intervals (10,000 draws, conditional on observed seeds). The primary contrast ($L_{\mathrm{BAL}} - L_{\mathrm{CE}}$ on replay data) is marked with an asterisk (*).

executing required coordination events (`WAIT` and `MERGE`), and obtaining an authenticated runtime certificate via `VERIFY` prior to `STOP`.

Primary estimand and uncertainty. Our primary estimand is the paired difference Δ = BAL_REPLAY − CE_REPLAY, averaged across both resource environments and the three training seeds. Statistical uncertainty is quantified via 10,000 cluster-bootstrap resamples over the 120 semantic bases, preserving task pairings across seeds and resource conditions. These intervals quantify task-sampling uncertainty conditional on the three observed training seeds.

6 Results

6.1 Balanced loss allocation improves closed-loop success

Figure 3 and Appendix Table A3 present the core experimental results. Across both data distributions, the balanced objective L_{BAL} substantially improved closed-loop performance over standard token-average cross-entropy L_{CE}. On the replay-replacement dataset, BAL_REPLAY achieved a mean task success of 94.17%, compared to 81.94% for CE_REPLAY, representing a paired improvement of 12.22 percentage points (95% bootstrap interval: [9.17, 15.42]). On expert demonstration data, BAL_EXPERT improved performance from 84.44% to 95.00%, yielding an advantage of 10.56 percentage points (95% interval: [7.78, 13.33]).

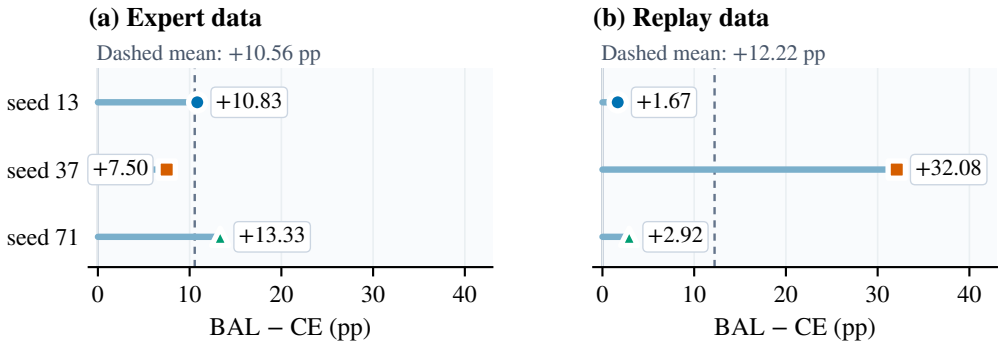


Figure 4: Seed-level gains from balanced loss allocation ($L_{\mathrm{BAL}} - L_{\mathrm{CE}}$). Points indicate paired performance improvements for each random seed, with dashed lines denoting overall condition means (+10.56 pp on expert data, +12.22 pp on replay data). Group normalization consistently improves performance across all seeds, resolving severe instability observed under standard cross-entropy on replay data (seed 37).

The reference baselines achieved 0/240 for BIND_NATIVE, 228/240 (95.00%) for BIND_TEACHER, and 158/240 (65.83%) for SCRIPT_INDEX. BIND_NATIVE's zero success confirmed that autonomous coordination did not emerge in the parent without control training. Autonomous BAL_EXPERT students matched the 95.00% success point estimate of the teacher-guided reference BIND_TEACHER, demonstrating that internalizing balanced control supervision enables unguided students to reach the performance level of scaffolded references.

6.2 State coverage alone does not substitute for balanced supervision

Comparing data regimes showed that offline replay replacement offered negligible advantage over expert demonstrations. Within CE, replay replacement yielded a difference of -2.50 points relative to expert data (95% interval: $[-5.97, 0.83]$); within BAL, the difference was -0.83 points (95% interval: $[-2.50, 0.83]$). The factorial interaction between loss formulation and data distribution was 1.67 points (95% interval: $[-1.81, 5.42]$), with marginal loss and data effects of 11.39 and -1.67 points respectively.

These findings showed that injecting recovery states into training data (D_1) without reweighting the objective failed to improve closed-loop execution. Indeed, under standard token-mean cross-entropy, the added complexity of recovery states slightly impaired performance. The primary determinant of performance in this regime is proper supervision allocation for control tokens rather than passive state coverage.

6.3 Group normalization stabilizes training across random seeds

Figure 4 details performance across individual training seeds. On expert data, L_{BAL} consistently improved performance across all three seeds (+10.83 pp for seed 13, +7.50 pp for seed 37, and +13.33 pp for seed 71). In the replay regime, standard cross-entropy displayed marked seed instability: while seeds 13 and 71 achieved 220/240 and 221/240, seed 37 dropped to 149/240 (62.08%).

L_{BAL} eliminated this instability, raising seed 37 to 226/240 (+32.08 pp) and maintaining consistently high success across all seeds (224, 226, and 228 out of 240). Seed 37 accounted for 77 of the 88 net additional successes in the replay contrast. Balanced loss allocation thus stabilizes fine-tuning against catastrophic control divergence.

6.4 Resource-sensitive failure modes and synchronization dynamics

Figure 5 stratifies closed-loop success by branch count and initial resource availability. In four-branch tasks ($n = 4$), introducing an initial idle worker elevated control complexity: CE_EXPERT dropped from 115/120 success on fresh-root problems to 57/120 when an idle worker was present.

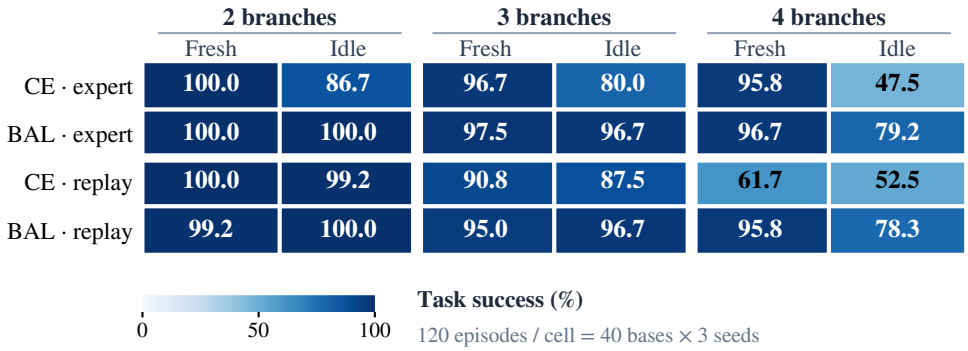


Figure 5: Task success rates (%) stratified by branch count ($n \in \{2, 3, 4\}$) and initial worker resource state. Each cell aggregates 120 episodes (40 semantic bases $\times$ 3 seeds). Fresh denotes a standard fresh root; idle denotes an initially idle compatible worker. Balanced supervision (L_{BAL}) maintains reliable coordination across increased concurrency ($n = 4$) and resource complexity, preventing premature synchronization failures.

In contrast, BAL_EXPERT maintained high success across both settings, completing 116/120 and 95/120 episodes respectively across the three seeds.

Diagnostic trace analysis identified the primary failure mechanism: under standard cross-entropy, policies frequently triggered premature synchronization. In CE_REPLAY/seed37 (which completed only 11/80 four-branch tasks), an adverse-cue analysis showed that 70 of 82 failure attempts stemmed from issuing `WAIT` before worker instances were fully assigned across all active branches. Group-normalized supervision prevented these premature control commitments: across paired seed 37 instances, BAL converted 79 CE failure cases into successful executions (with only 2 reverse shifts and 147 joint successes), confirming that balanced loss directly reinforces sequential control discipline.

6.5 Distinguishing Semantic Validity from Reference Preference

To evaluate decision flexibility, we distinguish exact agreement with reference actions from semantic validity. In counterfactual state evaluations, CE policies match the specific preferred reference opcode on 40 of 60 pairs, yet maintain 100% (60/60) runtime contract validity and productive task progress. Classifying these 20 alternative valid actions as failures would falsely conflate stylistic reference preference with operational efficacy. Our framework separates action preference from runtime validity, progress, and end-to-end task completion (Appendix Figure A1).

Appendix Figure A2 illustrates this principle via recorded autonomous BAL student traces. In the successful execution, the root policy allocates four parallel workers, awaits completion, verifies the integrated state, and terminates with an authenticated certificate. In the failure example, execution progresses smoothly through synchronization, faltering only at the final `MERGE` step due to an argument key mismatch. This granular tracing confirmed that students operated as end-to-end autonomous agents and provides concrete visibility into their execution mechanics.

6.6 Operational Boundaries Under Joint Tool-and-Control Shift

To delineate the generalization limits of frozen policies, we evaluate all six expert-data students alongside the three reference baselines on 48 held-out base problems (96 paired episodes) under a prospective joint shift. This follow-up replaces the branch arithmetic with list maxima or threshold counts and alters the root-level objective from dictionary collection to scalar reduction, necessitating an extended sequence of canonical root actions (`CALL` and `WRITE`).

Under this simultaneous tool-semantic and control-protocol shift, all nine frozen conditions achieved 0/96 terminal successes (Appendix Table A5). In contrast, deterministic CPU reference executions pass 192/192 paths, confirming that the extended task interface is computationally feasible. Detailed

termination diagnostics identify 523 consecutive no-progress terminations and 213 runtime exceptions. Because tool semantics and root-level control requirements change together, this finding maps an operational boundary for frozen policies under compounded interface shifts rather than isolating an individual failure component.

7 SCOPE, LIMITATIONS, AND FUTURE DIRECTIONS

Our findings characterize control and binding within a controlled procedural task family using an adapted Qwen3-4B parent. The evaluation protocol intentionally decouples policy learning from platform execution mechanisms: actor scheduling, typed permissions, and runtime certificate verification remain modular components of the host execution environment, ensuring that observed performance gains reflect learned policy discrimination. While the common parent was adapted on the declared 192 training tasks, evaluation inputs maintain verifiable separation from pilot distributions (Appendix B).

Equal loss weighting (L_{BAL} with fixed 0.5 group coefficients) provides an effective formulation that substantially outperforms standard token-average objectives. Future theoretical and empirical studies may explore continuous parameterizations ($\alpha \in [0, 1]$), adaptive gradient-balancing schedules, and interactions with learning rate dynamics to characterize optimization dynamics across varying training budgets.

Our single-pass offline replay substitution establishes that bounded recovery-state coverage cannot compensate for diluted control supervision within a fixed data budget. Combining group-normalized objectives with multi-round iterative collection (such as online DAgger rollouts) represents a natural next step to determine whether active on-policy exploration compounds with balanced supervision. Finally, fine-tuning and evaluation consumed 95.4435 allocated GPU-hours and 67,007 model requests, with the joint-shift follow-up accounting for 6.9616 hours and 8,261 requests. The follow-up test maps an operational boundary: frozen zero-shot policies cannot navigate compounded shifts in tool semantics and root-level aggregation protocols, confirming that structural interface shifts require targeted policy adaptation (Appendix E).

8 CONCLUSION

This work identifies and resolves a fundamental structural bottleneck in agent policy distillation: the sequence-length disparity between concise operational control codes and extensive argument bindings. Standard token-mean cross-entropy dilutes critical control supervision beneath lengthy argument payloads, leading to operational failures during autonomous execution. Through a controlled 2×2 factorial study with matched training exposure and optimizer updates, we show that restoring supervisory parity via an action-group-normalized objective (L_{BAL}) yields double-digit success gains (+12.22 percentage points on replay-replacement data, +10.56 points on expert data) across multiple seeds. Operating within a shared platform runtime, students select opcodes and arguments without external guidance, achieving 94.17% and 95.00% success and matching the 95.00% point estimate of a teacher-guided reference. Furthermore, decoupling supervisory allocation from the training-state distribution demonstrates that expanding offline recovery-state coverage cannot compensate for unbalanced loss allocation within fixed data budgets. By addressing the structural imbalance between control and argument supervision, this work shows that loss allocation governs autonomous policy competence and provides a clear operational framework for distilling self-directed language agents.

ETHICS STATEMENT

This controlled study uses procedural integer tasks rather than human-subject data. Execution occurs within a constrained tool interface, with common actor permissions and terminal verification. These findings concern that controlled setting, not deployment safety on unrestricted tools. Supplementary outcome tables omit task content, personal identifiers and private provenance paths.

AI USE STATEMENT

Generative AI tools were used exclusively for language polishing and writing assistance in drafting and revising the manuscript prose. All conceptual formulations, experimental designs, algorithmic implementations, analyses, and scientific conclusions were conducted and verified by the authors.

REPRODUCIBILITY STATEMENT

Appendix A specifies the recorded optimization settings; Appendices B and C describe the common parent and execution contract. Anonymous supplementary tables contain all 3,600 core and 864 follow-up outcomes with shared base aliases. A CPU-only entry point recomputes the original task-cluster statistics, including the resampling seeds and cluster order. It also checks the six factorial contrasts and seed sensitivities. Per-row objective coefficients, resource-stratified diagnostics, and de-identified executed traces accompany those tables. These materials support recomputation and verification from recorded outcomes, providing an audited record of the findings while training recipes and execution contracts enable reproduction on comparable hardware.

REFERENCES

Wei Chen and Zhiyuan Li. Octopus v2: On-device language model for super agent, 2024. URL https://arxiv.org/abs/2404.01744v3.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023. URL https://arxiv.org/abs/2305.14314.

Bingguang Hao, Maolin Wang, Zengzhuang Xu, Cunyin Peng, Yicheng Chen, Xiangyu Zhao, Jinjie Gu, and Chenyi Zhuang. FunReason: Enhancing large language models' function calling via self-refinement multiscale loss and automated data refinement, 2025. URL https://arxiv.org/abs/2505.20192v1.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016. URL https://arxiv.org/abs/1511.06295.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin Peng, Huan Zhang, Jianfeng Gao, and Tong Zhang. GUI-Libra: Training native GUI agents to reason and act with action-aware supervision and partially verifiable RL, 2026. URL https://arxiv.org/abs/2602.22190.