Recovery Policy Learning / Grounded Recovery

Budget-matched corrective imitation learning for recovery
in language-conditioned embodied policies in BabyAI/MiniGrid

Protocol 0.1.0 · frozen contract, one receipted test opening

1 Executive summary

A policy trained by behavioral cloning is fitted on the expert's states and then, at test time, has to act under its own. The moment it makes a mistake it is somewhere the demonstrations rarely went, and its error rate there is unmeasured. The standard remedy is to spend expert labels in the states the learner actually reaches. The standard alternative is simply to collect more demonstrations. Which is the better use of the same budget is an accounting question, and it is asked far less often than it is assumed.

This study asks it under strict matching. Given a fixed additional budget of 1000 revealed oracle labels, identical supervised target exposure, and identical optimization, is that budget better spent on **recovery-state aggregation**, meaning DAgger-style labels queried in states the learner visits after one corrupted action, or on **extra nominal demonstrations** of the kind the base policy already learned from?

The primary endpoint is eligible unseen one-corruption intention-to-treat success: closed-loop task success on a frozen, never-touched panel of 536 scenarios, in which a held-out corruption operator replaces one policy proposal per episode at a scheduled time. The estimate is the mean within-bundle paired difference across 6 complete pipeline replicates.

Recovery aggregation minus extra demonstrations = +14.3 percentage points
95% paired t interval [+10.0, +18.6] pp
analysis status: confirmatory · claim state: *support*

Three things about that number are worth stating immediately, because they are what a reader should check rather than take on trust.

First, it was not a completion criterion. The frozen protocol committed in advance to reporting a null, adverse, or inconclusive interval with exactly the same prominence, and the interpretation rule was written down before the panel was opened.

Second, the two arms really are matched. Both start from a bit-identical clone of the same base checkpoint, reveal the same number of oracle targets, and take the same number of optimizer updates on the same mix of old and new targets. What they cannot share is the cost of acquisition, and that difference is logged rather than called equal: the recovery arm made 38,109 oracle calls to reveal its labels against 6,000 for extra demonstrations, discarding 32,109 recommendations that fell outside its reveal window.

Third, the advantage is not confined to the corrupted slice. Recovery labels also beat extra demonstrations when no corruption is delivered at all, which points at the on-policy character of the labels rather than at the policy having learned to undo one specific operator.

The rest of Part I states the question precisely, describes what was built, reports the primary endpoint with every replicate point, then reports the trade-offs, the failures, two exploratory panels opened afterwards, and the boundary of what this evidence can support.

2 How to read this report

The report is in two parts, and they answer different needs.

Part I, the Grounded Recovery study, is the confirmatory experiment: the question, the frozen protocol, the system that implements it, the result, its trade-offs, and the limits of what it establishes. It is self-contained. A reader who wants the evidence and nothing else can read Part I and stop.

Part II, Foundations, is how the study was built up from nothing. Seven small deterministic side-studies construct and measure each ingredient in turn: the simulated world, the formal decision problem, the teacher that produces labels, the learning paradigm, the policy network, the failure mode that motivates the whole experiment, and the measurement discipline that makes the answer worth believing. Each is backed by its own runnable code in `src/gr_foundations/`, and each ends by pointing at the place in Part I that uses the idea.

The order is deliberate. Part I leads because the result is the point. Part II follows because a reader who wants to check a premise, or who is meeting partial observability or intention-to-treat accounting for the first time, should be able to descend into it rather than wade through it first. Part I cross-references Part II throughout, so any claim in the study can be traced down to the small experiment that establishes it.

> **Evidence status.** Everything in Part II is exploratory teaching material: small sample sizes, no frozen protocol, no confirmatory claims. All confirmatory evidence in this report lives in Part I, produced under the frozen contract and a single receipted test opening. Where Part I and Part II numbers appear to disagree, Part I is the evidence; Part II exists to make Part I auditable, not to add claims to it.

Every empirical number in either part is generated by code into `report/generated/` and imported here. None is typed by hand, which is why a re-analysis cannot leave a sentence quietly stale.

Contents

Chapters 3 to 14 are Part I, the confirmatory study. Chapters 15 to 22 are Part II, the foundations track, which is exploratory throughout.

Part I · The Grounded Recovery study

The confirmatory experiment, under the frozen contract,
with one receipted test opening.

3 Question and scope

3.1 The question in words

Suppose a policy has already been cloned from 4000 expert-labelled decisions and performs reasonably well. Someone offers you 1000 more oracle labels. You can spend them in two ways. You can ask the expert to demonstrate more tasks from scratch, which produces more of exactly the kind of data you already have. Or you can let your own policy act, corrupt one of its actions so that it ends up somewhere it would not normally be, and ask the expert what to do from there.

The second option is the DAgger family's answer to closed-loop distribution shift, and its motivation is well known. What is far less often measured is whether it actually wins when the two options are given the same budget, the same starting weights, the same number of gradient steps, and the same number of supervised targets per step. Without that matching, an apparent advantage can come from having seen more data, or trained longer, or started from a better checkpoint.

This study measures the comparison with all of those held fixed.

3.2 The estimand

For method m, pipeline bundle r, and eligible unseen scenario e, let $Y[m, r, e] \in \{0, 1\}$ be intention-to-treat task success. With per-policy rates $\hat{p}[m, r] = \mathrm{mean}_e Y[m, r, e]$, the primary contrast is

$$\delta_r = \hat{p}[\mathrm{recovery}, r] - \hat{p}[\text{extra demo}, r], \quad \hat{\Delta} = \mathrm{mean}_r \delta_r,$$

estimated with a prespecified 95% paired t interval across complete pipeline bundles, conditional on the frozen scenario panel. The smallest effect size of interest is 0.05 absolute success.

Three choices inside that definition carry most of the weight, and each is defended in its own chapter.

The **unit** is the pipeline bundle, not the episode. A bundle is one complete run from base data through base training, arm cloning, four rounds of collection and fine-tuning, to two final policies. Episodes inside a bundle are repeated measurements of the same trained policies, so treating them as independent replicates would shrink the interval by roughly the square root of the panel size and claim a precision the design does not have (*Measuring honestly*).

The **comparison** is recovery against extra demonstrations. The untouched base policy is reported throughout, but it is a context line, not the competitor. Beating a policy that received no additional labels at all would say nothing about how to spend a budget.

The **scoring** is intention to treat. A scenario assigned a corruption stays in the denominator whether or not the corruption was delivered, and whether or not the episode ended first (*Study design*).

3.3 In scope

A symbolic, discrete, fully scripted-oracle BabyAI study of label allocation under closed-loop distribution shift, with exact budget and exposure accounting, leakage-resistant scenario splits, and one receipted confirmatory test opening.

3.4 Out of scope

Reinforcement learning, in the strict sense that no reward term exists anywhere in the update; physical robots; continuous control; human supervision; pretrained perception or language models; and any claim that BabyAI evidence transfers as such to real robotics. The precise name for the treatment is **DAgger-style recovery-state aggregation**: adaptive data collection followed by masked behavioral cloning. It is neither the full DAgger algorithm nor reinforcement learning, and it is described that way throughout.

What the study can support, and what it deliberately cannot, is set out in *Discussion and limitations* rather than left to the reader to infer.

4 Background in three ideas

Part II develops each of these from the ground up with its own measurements (*When cloning breaks*, *Decision processes*, *Measuring honestly*). This chapter is the compressed version for a reader who starts here.

4.1 Imitation under covariate shift

Behavioral cloning treats control as supervised learning: minimize prediction error against expert actions on states the expert visited. The trouble is that the trained policy is then deployed on states **it** visits, and those are not the same distribution. One wrong action moves the agent somewhere the demonstrations covered thinly, its error rate there is higher and unmeasured, and the next state is drawn from an even worse distribution. Errors compound along a trajectory rather than averaging out across it, which is why a policy with excellent per-step accuracy can still fail the task.

The remedy family that DAgger introduced is to move the labels rather than to add more of them: query the expert in states the learner actually reaches, so that supervision covers the distribution the policy will face. The open accounting question, and the one this study asks, is whether relocating the labels beats simply buying more of the ordinary kind at the same price.

Naming matters here. This study aggregates oracle labels in learner-visited post-corruption states and fine-tunes on them. It does not implement DAgger's policy mixing or its regret analysis, so it is called **DAgger-style recovery-state aggregation** and not DAgger.

4.2 Partial observability and recurrence

The agent sees a 7×7 egocentric symbolic view of a maze of nine rooms, so the task is a partially observed Markov decision process. The optimal action depends on history, for instance on which rooms have already been searched, and not only on the current view. Part II measures this directly: distinct underlying states produce byte-identical observations for which the oracle recommends conflicting actions, so no memoryless policy can be optimal (*Decision processes*).

The policy therefore carries a recurrent state over the fused observation, instruction, and action-history features, and supervision is applied to history windows rather than to isolated frames. This is not an architectural preference; it is forced by the decision problem.

4.3 Intention-to-treat evaluation

Every scenario assigned a scheduled corruption stays in the denominator, including episodes that ended before the corruption could be delivered. The reasoning is the same as in clinical trials, and the bias it avoids is specific and easy to fall into.

Delivery is not an independent event. A weaker policy ends its episodes sooner, so a scheduled corruption is more often never reached, so its assignments are more often undelivered. Conditioning the analysis on delivery would therefore drop exactly the hardest episodes from the weakest arm and flatter it. Delivery is a post-treatment variable, and Part II demonstrates the resulting bias on simulations with known ground truth (*Measuring honestly*).

The cost of the honest choice is that the reported rates are slightly diluted by episodes where nothing was done to the policy at all. That dilution is reported rather than corrected: assigned and delivered counts appear side by side in *Results*.

5 Environment and oracle

The world, the teacher, and the operator family are each built from scratch and measured in *The world*, *Policies and oracles*, and *When cloning breaks*. This chapter states what the frozen contract fixes and why.

5.1 Environment

`BabyAI-GoToObjMazeS4-v0`: a 3×3 maze of 4×4 rooms with one distractor, templated missions of the form "go to a/the {color} {type}", a limit of 144 steps, and observations consisting of a 7×7×3 symbolic image plus a view direction and the mission string.

The contract sets the environment's own `doors_open` parameter to true, and this was a pilot discovery rather than a convenience. With closed doors the scripted oracle must emit `toggle` in order to pass through them. That action lies outside the frozen three-action set, so it would either enlarge the action set, and with it the whole corruption operator family, or make the oracle unsupportable at exactly the states where recovery matters most. With open doors the oracle is movement-only, and that is verified over every collected trajectory rather than assumed.

5.2 The frozen action set, and why it is frozen

The action set is exactly $\{\mathsf{left}, \mathsf{right}, \mathsf{forward}\}$.

A corruption operator has to change every action it touches, since an operator with a fixed point would silently turn some corruptions into no-ops. An operator that changes every element is a **derangement**. On a three-element set exactly two total derangements exist, the two 3-cycles

$$g_+ = (1, 2, 0), \quad g_- = (2, 0, 1),$$

and each is the other's inverse. One of them, g_+, is the collection operator. The other, g_-, paired with a disjoint set of scheduled times, is the unseen operator.

This is a hard structural fact, not a design choice, and it is pinned by an exhaustive enumeration test. It also sets a real limit on the study, which is stated plainly rather than buried: with only two derangements available, "unseen" cannot mean an independently sampled corruption family. It means the unique other derangement, delivered at times the collection operator never used. That limit is revisited in *Discussion and limitations*.

5.3 The synchronized oracle

The MiniGrid `BabyAIBot` is wrapped so that `replan(last_executed)` is called exactly once per active simulator step and always receives the action that was actually executed, never the policy's proposal. Double calls, skipped calls, calls after termination, and recommendations outside the frozen action set all raise rather than proceed.

The reason for this severity is that the failure it prevents is silent. If the bot is told a different action from the one the world took, it replans from a state that does not exist, and every label derived from that point onward is attached to the wrong state. Nothing crashes; the data simply becomes wrong. Part II attempts to falsify the wrapper by lying to it, skipping calls, and double-calling it, and finds that the bot's own replanning is

robust enough to recover anyway (*Policies and oracles*). The contract is therefore not fragility protection. It is accounting protection: it guarantees that a recorded label belongs to the state it is recorded against.

5.4 Operator preflight

Before any learner ever ran, both operator families were tested in isolation: 600 episodes per family, one forced corruption each, with the scripted oracle driving the recovery. The oracle recovered in all 1200 delivered corruptions, against a frozen gate of 0.99.

This gate exists to separate two very different explanations of a failure. If the learner fails to recover after a corruption, that is a fact about the learner. If the **oracle** could not recover either, the corruption was not a recoverable perturbation in the first place and the endpoint would be measuring task impossibility rather than policy quality. The preflight rules that explanation out in advance.

6 Study design

The design principles used here, meaning matched budgets, intention-to-treat scoring, and paired replication, are each demonstrated with known-truth simulations in *Measuring honestly*; the three-arm structure is previewed at small scale in *When cloning breaks*.

6.1 Three arms, one shared start

Each pipeline bundle trains one base policy on the shared dataset D_0 of exactly 4000 revealed targets, then clones it, parameters and optimizer state asserted bit-identical, into the two full-budget arms:

- **bc_base**: the untouched base checkpoint, zero post-base updates. A context line, not the competitor.
- **extra demonstrations**: 1000 additional oracle targets drawn from fresh nominal trajectories, 250 per round over 4 rounds.
- **recovery aggregation**: 1000 additional oracle targets drawn from the learner's own post-corruption states. The current policy rolls out, one scheduled proposal is corrupted to $g_+(\text{proposal})$ and executed, and the oracle's recommendations for at most 8 successive states after the corrupted transition are revealed, until the exact per-round budget is met.

Both arms are re-cloned from the **same** base checkpoint in every bundle, so a bundle contributes one paired difference in which the only thing that differs is where the labels came from.

6.2 What is matched, and what is only logged

After each round both arms train for exactly the same number of optimizer updates with identical base and new target slots per update. A fairness audit recounts the immutable ledgers and refuses to continue on any inequality in updates, drawn targets, or cumulative exposures. Over the 6 bundles that comes to 768,000 base target exposures, 192,000 new target exposures, and 12,000 optimizer updates for each of the two arms, identical on every count.

Quantities that cannot be matched are logged and reported rather than called equal. Acquiring recovery labels means running the learner, so most of the oracle recommendations produced along the way fall outside the reveal window and are discarded. Over the 6 bundles the extra-demonstration arm made 6,000 oracle calls and took 6,000 simulator steps to reveal its budget, while the recovery arm made 38,109 calls and discarded 32,109 recommendations to reveal the same number of labels. That is the honest price of the method, and it is quantified in *Results* rather than described qualitatively.

6.3 Evaluation slices

Three slices are evaluated, all crossed over the identical ordered panel with identical per-scenario schedules for every arm and bundle:

clean no corruption. Answers whether the added labels helped nominal behavior at all.

matched g_+ at the collection times. Answers whether the policy learned to handle the exact perturbation it was trained under. Findings here are perturbation-family-specific by construction.

unseen g_- at a disjoint time set. **Primary.** Answers whether the added labels bought anything beyond the operator and schedule that produced them.

6.4 Eligibility, decided before any outcome

A scenario enters the unseen panel only if the nominal oracle path is longer than the latest scheduled unseen corruption time. Otherwise the episode would be over before the corruption could be delivered, and the scenario would contribute a guaranteed non-event to every arm alike.

The filter uses only the fixed nominal oracle path length, a property of the scenario computed before any policy existed, and retains 536 of 800 test candidates. No learner output influences it. This is eligibility, decided in advance for all arms at once, and it is emphatically not an outcome-dependent exclusion.

6.5 The decision rule, written down first

The interpretation of the interval was fixed at freeze time, before the panel was opened:

state	**condition on the 95% paired *t* interval**
support	lower bound above zero
adverse	upper bound below zero
rule out	interval contained within ±SESOI
inconclusive	anything else

Confirmatory status additionally requires at least $\max(5, R_{\text{train}})$ complete bundles. The precision target is reported with a met or unmet flag and does not by itself change the status. Writing this down first is what makes the reported state a finding rather than a choice made after seeing the numbers.

7 Data representation and policy model

The architecture is dissected component by component, with a matched five-way ablation, in *The policy network*; the naive episode-level training that this pipeline tightens is developed in *Learning paradigms*.

7.1 Records

Each episode stores per-step arrays, meaning the symbolic image, the direction, the previous, proposed, recommended and executed actions, the reveal and perturbation flags, and the termination state, alongside a JSON sidecar with provenance digests and a content-addressed checksum over the array bytes. The checksum is over the arrays, never over the container bytes, so re-saving the same data cannot change its identity.

Collection appends to hash-chained ledgers, and an independent recount re-reads every stored episode before training and refuses to proceed on any disagreement. Keeping proposed, recommended, and executed as three distinct fields, rather than collapsing them, is what makes it possible to verify after the fact that a label was attached to the state the world was actually in.

7.2 Training items

One training item is a contiguous history window ending at exactly one revealed target, with the prefix capped at 32 steps. That definition is the reason equal-exposure accounting is mechanical rather than approximate: one sampled item is one target exposure, so counting items counts supervision.

The alternative, training on whole episodes, quietly breaks the comparison. A nominal demonstration is label-dense along its whole length, while a recovery window is a short burst of labels after a corruption, so an episode-level batch delivers systematically more new labels per update to the extra-demonstration arm. Part II leaves exactly that leak open on purpose and measures it (*When cloning breaks*); the study closes it with the one-target-per-window rule.

Two details prevent off-by-one errors from becoming silent label corruption. The START token exists only at absolute episode start, so a window clipped by the prefix cap begins with the true executed action that preceded it rather than with a fake start. Padding exists only inside collation and is masked out of the loss, so a padded batch and an unpadded one produce the same gradients.

7.3 Model

Separate embeddings for the object, color, and state channels feed two 3×3 convolutions and a linear projection. A word-embedding GRU encodes the mission. The previous **executed** action and the view direction are embedded. A linear fusion and one policy GRU produce one logit per frozen action. The whole network is 179507 parameters, which is small enough to audit by hand and is asserted against the generated shape walkthrough in *The policy network* rather than quoted from memory.

The `forward` signature admits only these inputs. Coordinates, goal positions, oracle state, and arm identity have no parameter to arrive through, and a signature test enforces it. The boundary is the type signature itself rather than a convention, which is the only version of that guarantee that cannot erode.

The loss is masked cross-entropy at target positions. Training uses AdamW, global-norm gradient clipping, and with-replacement sampling from named-seed generators. All model computation runs on the contract's pinned device, and run-to-run bit-determinism of the resulting model state is revalidated per device rather than assumed to hold across hardware.

8 Verification and reproducibility

Each mechanism in this chapter is demonstrated in miniature, on cases with known ground truth, in *Measuring honestly*.

The test suite is an executable statement of the scientific contract rather than a safety net for refactoring. The invariants below are chosen for what their failure would do to the evidence, not for how hard they were to write.

Oracle synchronization exactly one `replan` per active step, fed the action that was executed. Failure would attach labels to states the world was never in, and nothing would crash.

Exact budgets base data holds exactly N_0 targets and each arm exactly B, with deterministic handling of a partial final window. Failure would break the budget-matched comparison that the whole study rests on.

Clone equality and fairness bit-identical arm starts; equal updates, target draws, and cumulative exposures at every round boundary; an unchanged base digest after arm training. Failure would let an advantage come from optimization rather than from allocation.

Causality logits at step t are unaffected by inputs after t, padding is inert, and stepwise inference reproduces the training-time forward pass exactly. Failure would mean evaluating a different function from the one trained.

Determinism the complete pipeline, from collection through training to checkpoints, reproduces bit-identical model state digests under the seed bundle.

Lifecycle freezing refuses unresolved placeholders and double freezes; final commands accept only the frozen contract; exactly one receipted confirmatory opening exists per contract; and the release-phase integrity check recomputes the published summary from the raw episode rows rather than trusting it.

Scenario manifests are hashed over the reset world content rather than over the seed, so two different seeds that happen to generate the same world cannot land in two different splits and leak. All eight purpose splits are pairwise disjoint by seed and by world content. Evaluation code never constructs the oracle, so no oracle recommendation can reach a policy at test time.

Six of those eight splits carry the experiment: base data, recovery collection, validation, operator preflight, the test candidate panel, and visualization. The remaining two, `difficulty_shift` and `expert_diagnostic`,

were reserved at freeze time for optional descriptive diagnostics and were not part of the confirmatory opening. They were opened afterwards, and what they show is in *Exploratory extensions*, labelled exploratory throughout. No split is generated and then left unaccounted for.

The one-opening rule deserves its own sentence. The confirmatory evaluator writes an append-only receipt, containing the panel identity and every checkpoint digest, **before** it reads a single outcome. An interrupted run may resume the same opening only from byte-identical inputs. There is no discretionary reopening and no selective rerun of an inconvenient cell. This is a process guard rather than a security claim: it does not make tampering impossible, it makes tampering something that has to be done deliberately and leaves a record.

9 Results

Everything in this chapter recomputes from the stored episode rows of the single confirmatory opening. The primary endpoint comes first regardless of its sign, followed by the secondary slices, the audits that make the comparison checkable, and the costs that the design could not equalize.

9.1 The primary endpoint

Across 6 complete pipeline bundles on the frozen eligible unseen panel, with 536 scenarios per cell and intention-to-treat scoring, the mean paired difference in one-corruption success, recovery aggregation minus extra demonstrations, is

$$\hat{\Delta} = +14.30 \text{ pp}, \quad 95\% \text{ paired } t \text{ interval } [+10.02, +18.59] \text{ pp}.$$

Applying the interpretation rule fixed at freeze time (*Study design*) gives claim state **support** with analysis status **confirmatory**. The interval lies entirely above the smallest effect size of interest, so the result is not merely distinguishable from zero but larger than the threshold declared worth caring about in advance.

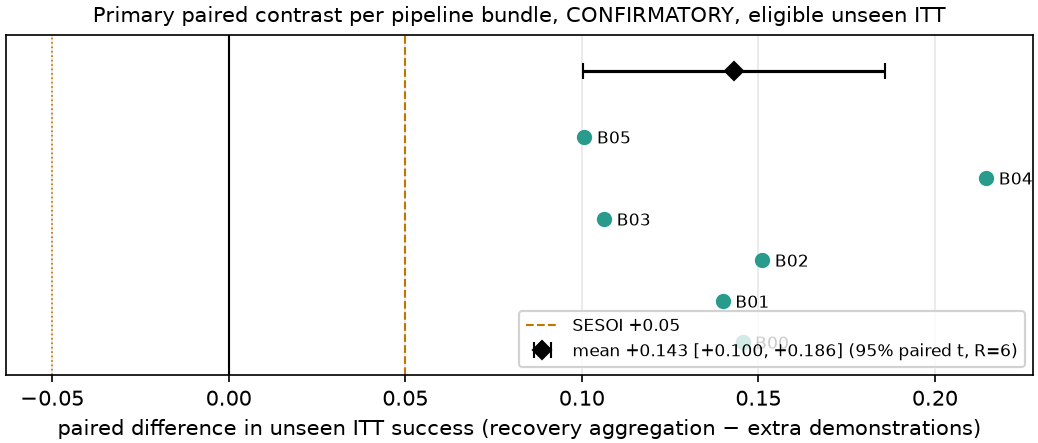

Figure 1: Primary paired contrast per pipeline bundle on the eligible unseen intention-to-treat slice. 536 scenarios per cell, 6 bundles, mean with a 95% paired t interval; the dashed line marks the smallest effect size of interest at 0.05.

Every bundle contributes one point, and 6 of 6 are positive:

bundle	paired difference (pp)
B00	+14.55
B01	+13.99
B02	+15.11
B03	+10.63

bundle	paired difference (pp)
B04	+21.46
B05	+10.07

The crossed two-way cluster bootstrap, with 2000 replicates and common bundle and scenario draws across arms, gives [+10.63, +18.35] pp. It is a labelled sensitivity analysis, not the primary interval, and it cannot compensate for the small number of pipeline bundles, because it resamples the same 6 trained pipelines. The prespecified precision target of a half-width at or below 0.05 was met, with an achieved half-width of 0.0428.

9.2 Success by slice and arm

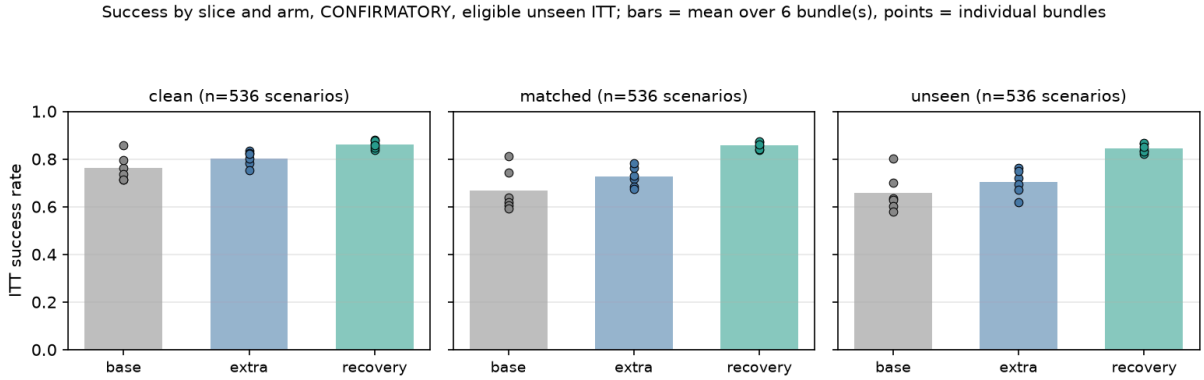

Figure 2: Intention-to-treat success by slice and arm. Bars are means over 6 bundles, points are individual bundles, denominators are in the panel titles.

arm	clean	matched	unseen (primary)
BC base	76.3%	66.9%	65.8%
Extra demonstrations	80.3%	72.5%	70.2%
Recovery aggregation	86.1%	85.7%	84.5%

Mean over 6 bundles, 536 assigned scenarios per cell, intention to treat.

Two features of this table matter more than the headline.

First, the corruption bites. The base and extra-demonstration arms lose roughly ten points when a held-out corruption is delivered, relative to their own clean rates, while the recovery arm loses 1.5 points. That is the effect the study was designed to detect, and it is visible without any statistics.

Second, the recovery arm also leads on the **clean** slice, where no corruption is delivered at all. The benefit is therefore not purchased with a loss of nominal performance, which is the trade-off one might have expected. It also suggests the mechanism is broader than "the policy learned to undo this operator", a reading taken up in *Discussion and limitations*.

9.3 Secondary paired contrasts

SECONDARY, NOT PRESPECIFIED

These contrasts use the same statistical unit as the primary endpoint and differ only in which arms and which slice they compare. They were not prespecified and are reported as secondary.

contrast	slice	mean (pp)	95% paired *t* (pp)
recovery − extra	clean	+5.78	[+3.67, +7.90]
recovery − extra	matched	+13.18	[+9.23, +17.14]

contrast	slice	mean (pp)	95% paired *t* (pp)
extra − base	unseen	+4.48	[−0.67, +9.63]
recovery − base	unseen	+18.78	[+11.13, +26.44]

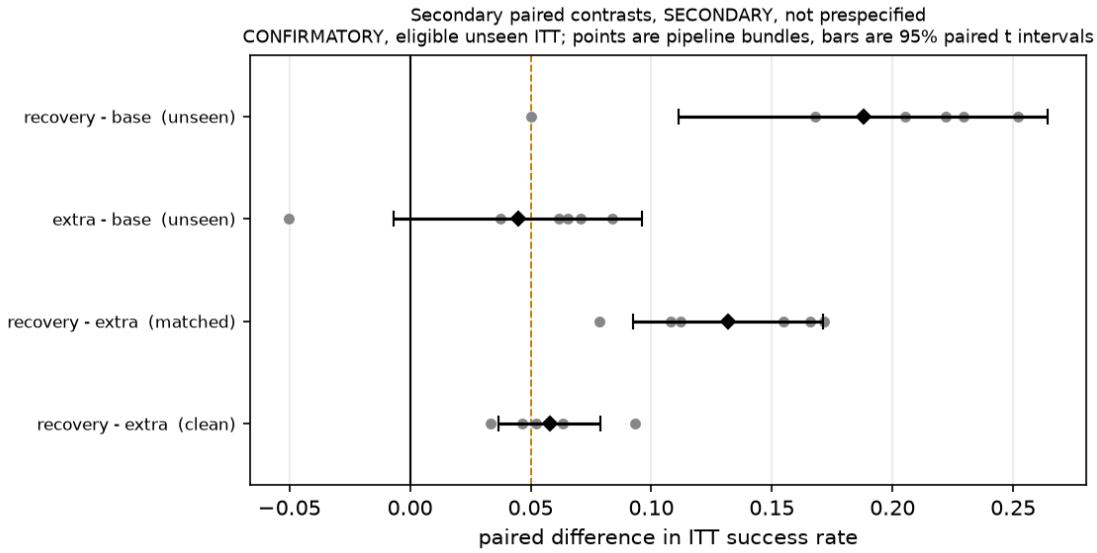

Figure 3: Secondary paired contrasts with every pipeline-bundle point. Diamonds are means with 95% paired *t* intervals; the dashed line marks the smallest effect size of interest.

The last row is the one a sceptical reader should look at first. The extra-demonstration arm, the study's actual competitor, is not clearly separated from the untouched base policy at this replicate count: its interval against the base includes zero. Extra demonstrations plausibly helped, but with 6 bundles the design cannot establish it. This does not weaken the primary comparison, which is paired within bundles and far better resolved, but it does bound how much should be read into the base line, and it is stated here rather than left for someone else to compute.

9.4 Assignment against delivery

Intention-to-treat scoring only means something if the gap between assignment and delivery is visible. On the unseen slice, of 3,216 assigned episodes per arm pooled over bundles, the corruption failed to land in 6, 4, and 6 cases for the base, extra-demonstration, and recovery arms respectively. On the matched slice the counts are larger, because those corruptions are scheduled later. Every one of these assignments stays in its denominator.

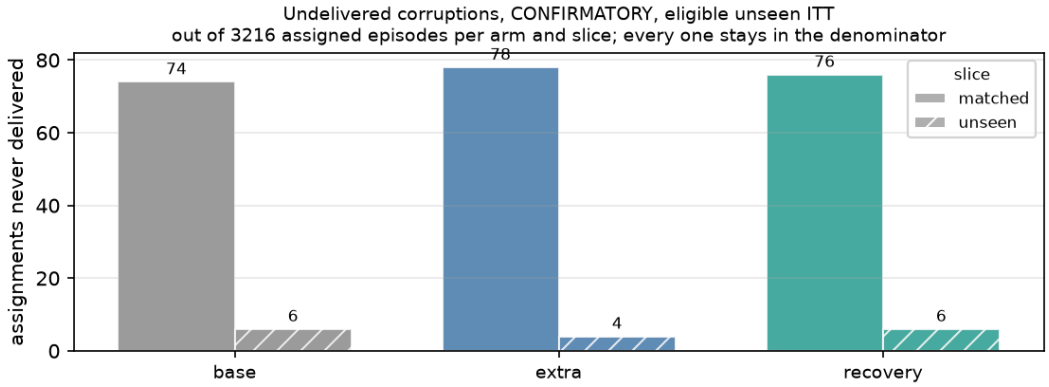

Figure 4: Assignments whose scheduled corruption never landed, pooled over bundles. Plotting delivered against assigned counts would show two nearly identical bars and hide the only quantity intention-to-treat accounting turns on.

9.5 Robustness across the corruption schedule

The unseen operator is scheduled at one of three times per scenario, fixed by the contract. Splitting the primary slice by that time shows the advantage is not an artifact of one particular moment of failure:

arm	t = 3	t = 5	t = 7
BC base	64.2%	65.5%	68.1%
Extra demonstrations	67.5%	71.4%	72.3%
Recovery aggregation	84.9%	83.1%	86.0%

Success rate on the unseen slice by scheduled corruption time, pooled over 6 bundles.

9.6 The budget and exposure audit

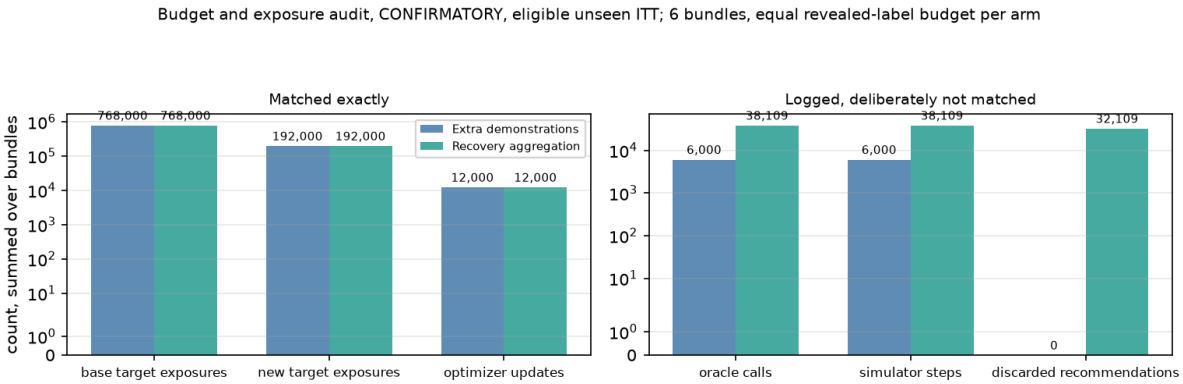

Figure 5: What the two full-budget arms share exactly, beside what they do not. Counts are summed over 6 bundles on a symmetric log scale, with exact values printed above each bar.

The left panel is the claim of fairness made checkable. Base target exposures, new target exposures, and optimizer updates are identical between the two arms, recounted from the hash-chained ledgers rather than asserted from the configuration.

The right panel is the cost the design deliberately did not equalize. Revealing 1000 recovery labels required 38,109 oracle calls and 38,109 simulator steps, against 6,000 for the same number of nominal demonstration labels, and discarded 32,109 recommendations that fell outside the reveal window. Recovery labels are therefore about 6.4 times more expensive to acquire in oracle queries. In a setting where the teacher is a scripted bot this is bookkeeping; in a setting where the teacher is a person it would be the dominant term, and any transfer of this result has to carry that number with it.

9.7 Success-conditioned path cost

A method that succeeds more often can still take longer paths when it does succeed, so the comparison is reported conditioned on success, with its denominator.

arm	successes	median step ratio	mean step ratio
BC base	2,115 / 3,216	1.261	1.523
Extra demonstrations	2,259 / 3,216	1.250	1.529
Recovery aggregation	2,719 / 3,216	1.324	1.583

Steps taken relative to the nominal oracle path length, on the unseen slice, among successful episodes only.

The recovery arm's successful episodes are modestly **longer** relative to the oracle path than either comparison arm's. This is a real trade-off and it runs against the headline: recovery buys a large gain in whether the task is solved at a small cost in how directly it is solved. It is reported here, beside the benefit, rather than in an appendix. The right panel of Figure 6 shows the full distributions.

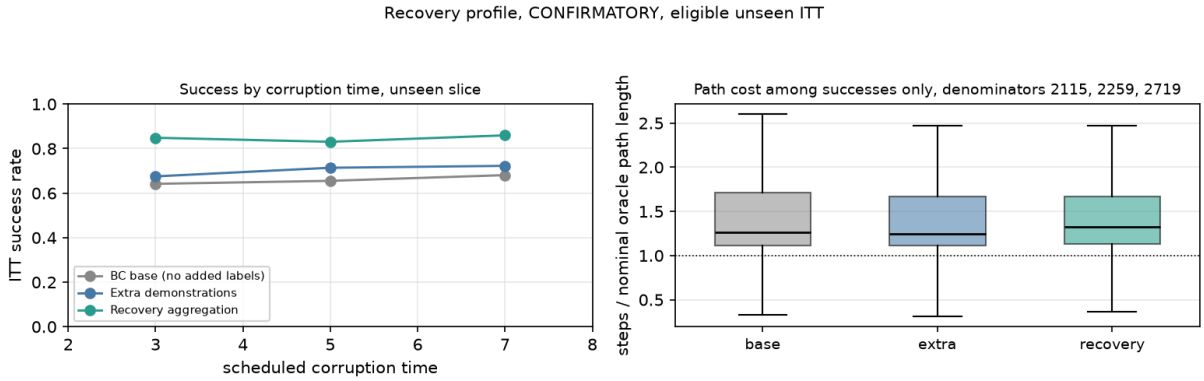

Figure 6: Left: success by scheduled corruption time on the unseen slice. Right: path cost relative to the oracle among successful episodes only, with denominators in the panel title.

9.8 Per-cell counts

The full crossed evaluation is 54 cells: 6 bundles by three arms by three slices. Every cell is reported, as successes over assigned episodes, with the number of delivered corruptions in parentheses.

bundle	arm	clean	matched	unseen
B00	base	408 / 536	342 / 536 (526)	341 / 536 (536)
B00	extra	447 / 536	384 / 536 (523)	386 / 536 (536)
B00	recovery	472 / 536	467 / 536 (522)	464 / 536 (536)
B01	base	382 / 536	331 / 536 (525)	337 / 536 (534)
B01	extra	419 / 536	366 / 536 (526)	372 / 536 (534)
B01	recovery	450 / 536	458 / 536 (526)	447 / 536 (535)
B02	base	395 / 536	325 / 536 (524)	322 / 536 (535)
B02	extra	429 / 536	392 / 536 (523)	360 / 536 (536)
B02	recovery	463 / 536	450 / 536 (523)	441 / 536 (536)
B03	base	427 / 536	399 / 536 (522)	375 / 536 (535)
B03	extra	442 / 536	409 / 536 (520)	408 / 536 (535)
B03	recovery	470 / 536	469 / 536 (525)	465 / 536 (534)
B04	base	382 / 536	318 / 536 (523)	311 / 536 (535)
B04	extra	404 / 536	362 / 536 (524)	331 / 536 (536)

bundle	arm	clean	matched	unseen
B04	recovery	454 / 536	451 / 536 (523)	446 / 536 (535)
B05	base	459 / 536	435 / 536 (522)	429 / 536 (535)
B05	extra	441 / 536	419 / 536 (522)	402 / 536 (535)
B05	recovery	459 / 536	461 / 536 (521)	456 / 536 (534)

10 Failure analysis

A success rate says how often a policy won. It says nothing about how it lost, and in a closed-loop study that second question is often the more informative one.

10.1 There is only one way to fail here

Across the entire confirmatory opening, every failure is a step-limit truncation. The count of episodes that terminated without reaching the goal is 0.

That is a degenerate composition, and reporting it is the point. In this environment there is no lava, no irreversible action, and no way to end an episode early except by succeeding: with the frozen three-action set and open doors, an agent that has gone wrong simply keeps moving until the 144-step limit expires. So "failure" throughout this report means one specific thing, a policy that never recovered its way to the goal within the budget, and not a mixture of failure modes that happens to average out.

This also means a whole class of question cannot be asked of this data. There is no catastrophic-versus-recoverable distinction to measure, because nothing here is catastrophic. A setting with irreversible failures would very likely change the balance between the two arms, and that is one of the sharper limits on transferring the result (*Discussion and limitations*).

arm	reached the goal	hit the step limit	ended without the goal
BC base	2,115	1,101	0
Extra demonstrations	2,259	957	0
Recovery aggregation	2,719	497	0

Terminal outcomes on the unseen slice, pooled over 6 bundles.

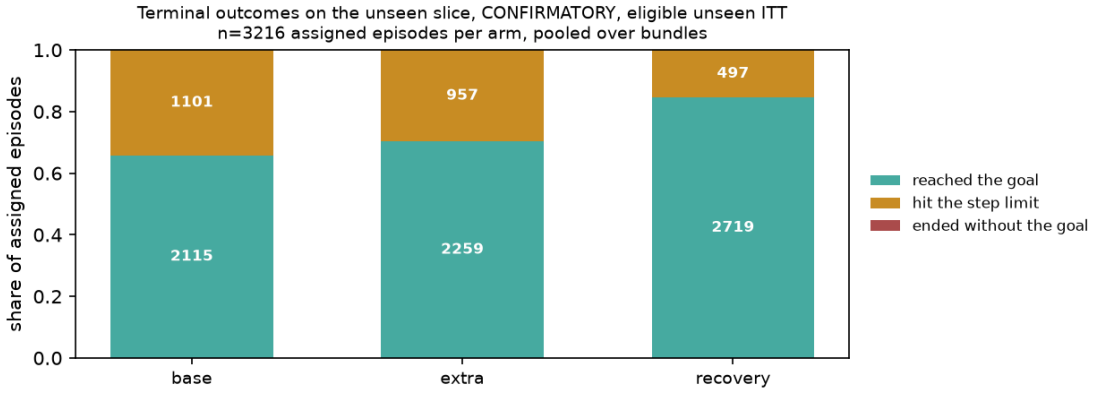

Figure 7: Terminal outcome composition per arm on the unseen slice. The third category is empty by construction, not by omission.

10.2 Where the recovery arm still loses

The recovery arm times out on 497 of 3,216 assigned unseen episodes. It is better than the alternatives, not close to solved: about one assigned episode in 6 still ends at the step limit.

The published media bundle therefore contains a recovery-arm **failure** as well as a recovery-arm success, both selected by a disclosed ordinal rule rather than by appearance: the paired contrast animation is the smallest eligible unseen scenario where recovery succeeded and extra demonstrations failed, and the failure animation is the smallest ordinal where the recovery arm failed with a delivered corruption. Each replay is deterministic and its outcome is asserted against the stored evaluation row before the file is written, so the animation cannot drift from the evidence it illustrates.

10.3 What the failures are not

They are not undelivered corruptions. Delivery was near universal on the unseen slice (*Results*), and the handful of undelivered assignments remain in every denominator, so they slightly **depress** every arm's rate rather than inflating any of them.

They are not eligibility artifacts either. The eligibility filter removed scenarios whose nominal oracle path was too short for the corruption to be delivered at all, using a property of the scenario computed before any policy existed, and it removed them identically for all three arms.

11 Exploratory extensions

EXPLORATORY, NOT PRESPECIFIED, OPENED AFTER THE CONFIRMATORY RESULT

Eight scenario splits were generated, hashed, and frozen before the study ran. Six of them carried the experiment. The remaining two, `expert_diagnostic` and `difficulty_shift`, were reserved for optional descriptive diagnostics and were not part of the confirmatory opening.

They are opened here, after the confirmatory result, and everything in this chapter is exploratory: not prespecified, not part of any claim, and recorded with its own opening note so that a post-hoc opening is itself on the record. Neither panel can move the primary estimand, and neither is used to support it.

11.1 Expert agreement is not closed-loop success

Each arm's final policy was run closed loop with no corruption on 200 scenarios per bundle, while the scripted oracle ran beside it in lockstep. At every step the policy's greedy action was compared with the oracle's recommendation for that same state.

arm	agrees with the oracle, per step	reaches the goal, per episode
BC base	41.6%	79.9%
Extra demonstrations	48.5%	84.8%
Recovery aggregation	66.8%	89.3%

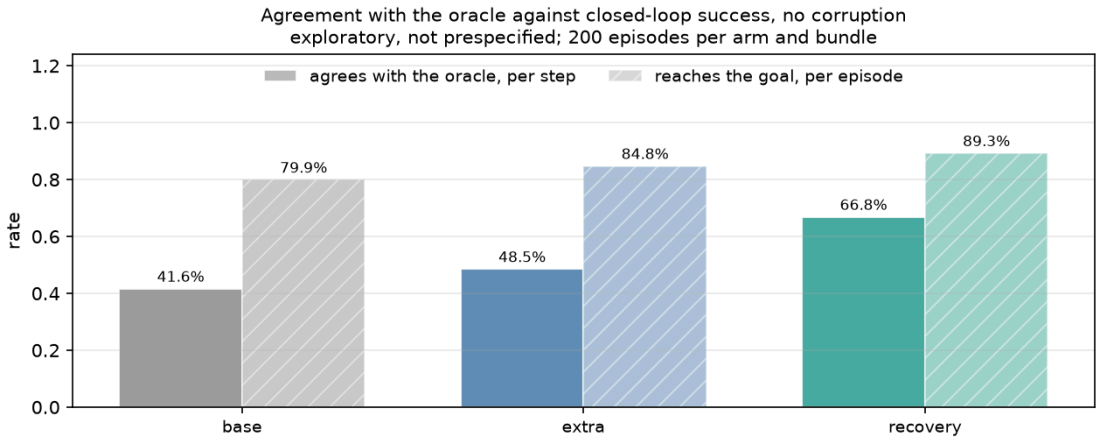

Figure 8: Per-step agreement with the scripted oracle beside per-episode closed-loop success, with no corruption delivered.

The gap between the two columns is the finding. The base policy matches the oracle's exact action on fewer than half of its steps and still reaches the goal four times in five. Agreement is a poor proxy for the thing the study actually measures, because in an open maze many actions are equally valid: two routes around a wall differ at every step and arrive at the same place.

This is the concrete reason the primary endpoint is closed-loop task success rather than action accuracy. A study that optimized or reported agreement would be measuring conformity to one particular expert trajectory, not competence.

The ordering is nonetheless worth noting: the recovery arm agrees with the oracle substantially more often, which is consistent with its labels having come from states the policy itself reaches. It is a descriptive observation here, with no interval and no claim attached.

11.2 Two corruptions instead of one

The primary endpoint is explicitly a one-corruption endpoint. This panel schedules **two** held-out corruptions per episode, at distinct times drawn from the unseen time set, over 200 scenarios per cell, and asks whether the advantage survives compounding.

arm	ITT success, two corruptions
BC base	69.9%
Extra demonstrations	75.4%
Recovery aggregation	87.4%

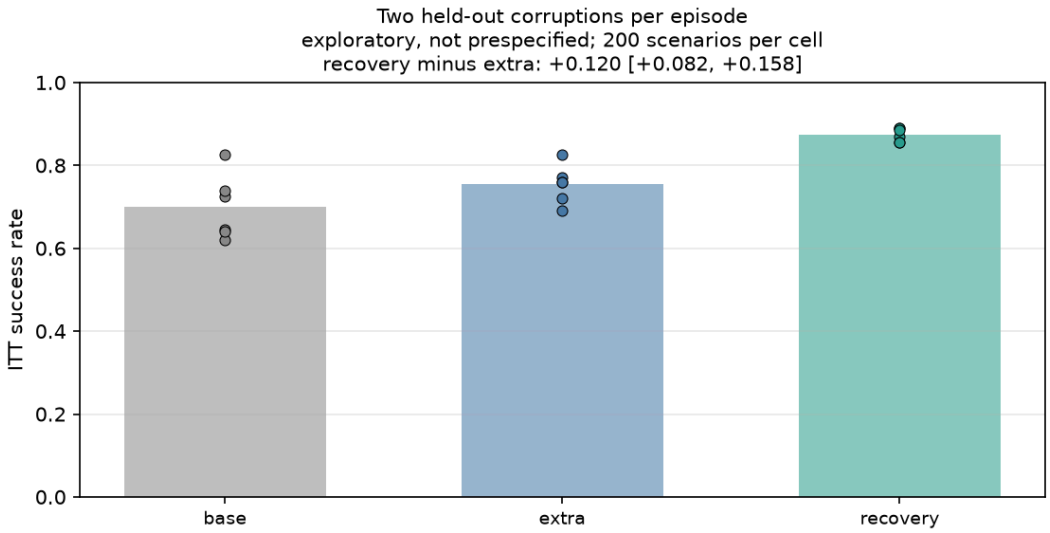

Figure 9: Intention-to-treat success under two held-out corruptions per episode. Bars are means over 6 bundles, points are individual bundles.

The paired recovery-minus-extra difference is +12.0 pp [+8.2, +15.8] pp, computed on the same statistical unit as the primary endpoint. The advantage persists under a harder perturbation, and is somewhat smaller than under a single corruption.

Scoring is intention to treat here as well, and it matters more on this panel than on the confirmatory one: roughly one episode in eight ends before its second scheduled corruption can be delivered. Those assignments stay in the denominator, and the per-cell delivered counts are in `exploratory/tables/two_corruption.csv`.

Read this as a direction rather than a measurement. The panel is a fifth the size of the confirmatory one, it was opened after the result was known, and its interval is descriptive. What it does rule out is one specific worry, that the recovery arm's advantage depends on the perturbation being exactly the single-corruption event it was evaluated under.

12 Discussion and limitations

12.1 What the comparison isolates

The design isolates label **allocation**. Both full-budget arms start from bit-identical checkpoints, reveal exactly the same number of oracle targets, and receive exactly the same optimizer updates on the same mix of old and new targets. What differs is only **where** the labels came from: states on fresh expert trajectories, or states the learner itself reached after a corrupted action. A difference on the unseen slice is therefore attributable to allocation, conditional on this environment, oracle, budget, and schedule.

That conditional clause is doing real work, and the rest of this chapter is about it.

12.2 The clean-slice result changes the reading

The recovery arm leads by +5.8 pp on the **clean** slice, where no corruption is delivered at all, with an interval excluding zero.

The obvious story for this study would be that the recovery arm learned to undo a particular corruption. The clean-slice result argues against that story. If the advantage were operator-specific, it should vanish when no operator is applied. Instead it persists, which points at the labels themselves: recovery labels are collected in states the policy actually visits, including the ordinary states it passes through in the eight steps after a corruption, so the supervision covers the deployment distribution better regardless of whether a corruption occurs.

That is the covariate-shift argument in its general form rather than a perturbation-specific repair, and it is a stronger reading of the result than the one the study set out to test. It is also, being a secondary contrast, not prespecified, and it should be treated as a well-supported observation rather than a confirmed claim.

12.3 Trade-offs, stated beside the benefit

Acquisition cost revealing 1000 recovery labels took 38,109 oracle calls against 6,000 for the same number of demonstration labels, discarding 32,109 recommendations along the way. The budgets that were matched are label budgets, not teacher-time budgets. With a human teacher the ranking could plausibly reverse, and nothing here tests that.

Path cost among successful episodes the recovery arm's paths are modestly longer relative to the oracle than either comparison arm's (*Results*). The gain is in whether the task is solved, at a small cost in how directly.

Failed rollouts are still consumed recovery collection runs the learner, and episodes that fail or truncate are stored rather than discarded, but they still consumed simulator time that the extra-demonstration arm did not spend.

12.4 Limitations

1. **The evidence is in silico, symbolic, discrete, and oracle-supervised.** Nothing here demonstrates physical-robot, continuous-control, or human-teacher competence. The teacher is a scripted planner with privileged access to the full grid, which is exactly the kind of teacher real robotics does not have.
2. **"Unseen" is narrower than the word suggests.** On a three-action set exactly two total derangements exist, so the held-out operator is necessarily the inverse of the collection operator (*Environment and oracle*). Unseen-ness rests on the distinct operator together with a disjoint set of scheduled times, not on an independently sampled corruption family. This is a structural consequence of the action set, disclosed rather than sampled away, and it is the single largest qualification on the headline.
3. **Every failure is a timeout.** The environment contains no irreversible action, so the study cannot say anything about recovery from a mistake that cannot be undone (*Failure analysis*). That is precisely the case where recovery labels would matter most in a physical system.
4. **The base policy was more competent than planned.** Perturbed competence sat near 65.8% rather than in the planned 20 to 70 percent range, because a single corrupted action is cheap to recover from in an open maze. The non-saturation criterion of at least ten points of headroom still held, and the deviation is recorded in the pilot report rather than smoothed over.
5. **The replicate count is small, by design.** The paired t interval is conditional on the frozen scenario panel and rests on 6 pipeline replicates, which is compute-bound rather than principled. Every per-bundle point is shown so that the reader can see the dispersion rather than infer it, and the extra-demonstration arm's own contrast against the base is not resolved at this count (*Results*).
6. **One environment, one task family.** All evidence comes from a single BabyAI configuration with templated "go to" missions. Nothing here establishes that the ranking holds for longer-horizon or compositional tasks.

What this study does not establish. That recovery-state labels beat extra demonstrations in general; that the result transfers to continuous control, real perception, or human teachers; that the comparison holds when teacher time rather than label count is the budget; or that any of this bears on reinforcement learning, which appears nowhere in the method.

13 Outlook

The point of naming the limits precisely is that each one implies a specific next experiment rather than a general aspiration. Four follow.

Budget the teacher, not the labels. The clearest weakness of the result is that it matches label counts while the arms differ roughly sixfold in oracle queries (*Results*). Rerunning the same design with oracle **calls** as the budgeted currency would answer a different and, for any setting with a human or an expensive teacher, more relevant question. The apparatus already logs the quantity that would become the constraint, so this is a protocol change rather than a rebuild.

Introduce an irreversible failure. Every failure in this study is a timeout (*Failure analysis*), which removes the case where recovery supervision should matter most. A variant with a genuinely absorbing failure state would test whether the advantage grows, as the covariate-shift argument predicts, or collapses because recovery labels arrive too late to help.

Break the derangement bottleneck. With three actions there are only two corruption operators, so "unseen" is structurally close to "matched". A larger action set would allow a held-out operator drawn from a genuinely different family, and would separate two explanations that this study cannot: whether recovery labels teach a general repair behavior or a specific inverse.

Test the clean-slice reading directly. If the advantage really comes from on-policy coverage rather than perturbation repair (*Discussion and limitations*), then labels collected in learner-visited states **without** any corruption should capture much of it. That is a third arm, cheap to add to the existing apparatus, and it would turn a well-supported secondary observation into a prespecified test.

None of these requires new machinery. The contract, the ledgers, the one-opening discipline, and the analysis path are all in place; what each variant needs is a new frozen protocol version and its own single opening.

14 Reproduction

The complete pipeline is deterministic given the frozen contract. From a clean checkout with `uv`, the pinned toolchain, and a system `ffmpeg` (needed only for the rollout animations):

```
uv sync --frozen
uv run gr smoke          --config configs/pilot.yaml
uv run gr make-manifests --config configs/pilot.yaml
uv run gr preflight      --config configs/pilot.yaml
uv run gr freeze         --config configs/pilot.yaml
for b in B00 B01 B02 B03 B04 B05; do
  uv run gr run-bundle --contract configs/experiment_contract.yaml --bundle $b
done
uv run gr integrity      --contract configs/experiment_contract.yaml --phase preopen
uv run gr evaluate-final --contract configs/experiment_contract.yaml
uv run gr analyze        --contract configs/experiment_contract.yaml
uv run gr audit          --contract configs/experiment_contract.yaml
uv run grf study-extras
uv run gr integrity      --contract configs/experiment_contract.yaml --phase release
uv run gr publish-result --contract configs/experiment_contract.yaml
uv run gr export-typst   --contract configs/experiment_contract.yaml
typst compile report/main.typ build/recovery-policy-learning-report.pdf
```

Named seeds are derived with SHA-256 from the root seed, the bundle identifier, and a component name drawn from a closed registry, so a typo produces an error rather than a quietly different random stream. The frozen contract, the code surface, the manifests, the eligible panel, and every released artifact carry content digests, recorded in `configs/freeze_record.json` and the artifact manifests. The release-phase integrity command re-derives the published summary from the raw episode rows rather than trusting the stored file.

Pinned versions: Python 3.11, `torch` 2.13.0, `minigrid` 3.1.0, `gymnasium` 1.3.0, with exact resolutions in `uv.lock`.

Part II runs separately as `uv run grf run all`, deterministically, under a seed namespace disjoint from the study's.

14.1 Provenance, stated exactly

The code digest recorded at freeze time identifies the source tree that produced the confirmatory result. After the test opening, the following additions were made, none of them scientific: the rollout-media generator together with a rendering accessor on the environment session, which is presentation plumbing that changes no contract semantics; the foundations package of Part II with its journey media and site-staging commands; the descriptive audits of *Results* and the exploratory panels of *Exploratory extensions*, which read the already-stored episode rows and write into their own files; and the website, which consumes only the staged evidence bundles.

The shipped tree therefore **does not** reproduce the freeze-time code digest, and it is worth saying so rather than leaving a reader to discover it. What verifies the result is not a match between the current source and a recorded digest; it is that the published summary recomputes, to numerical tolerance, from the immutable episode rows of the single opening, which is what `gr integrity --phase release` checks and what any reader can rerun. No frozen protocol value, manifest, checkpoint, or confirmatory result artifact was modified at any point.

Part II · Foundations

Every ingredient of the study, built from the ground up and measured:
the world, the decision problem, the teacher, the learning paradigm,
the network, the failure mode, and the measurement discipline.

Exploratory teaching material. All confirmatory evidence lives in Part I.

15 The foundations track

Part I reported the experiment. This part is how it was built up from nothing.

Seven small, deterministic, tested side-studies construct each ingredient in turn and measure it. The code lives in `src/gr_foundations/` and runs with `grf run all`; every empirical number below is generated by that code into `report/generated/foundations/` and imported here, so nothing in this part is typed by hand either. Each chapter ends with a bridge that names the place in Part I where the idea is used.

Evidence status. Everything in this part is exploratory teaching material: small sample sizes, no frozen protocol, no confirmatory claims. All confirmatory evidence in this report lives in Part I, produced under the frozen contract and a single receipted test opening. Where the two parts appear to disagree, Part I is the evidence; this part exists to make Part I auditable, not to add claims to it.

The route. Seven questions, in dependency order:

chapter	question answered	key artifact
The world	What is BabyAI, and what does the agent actually see and do?	observation anatomy; world census
Decision processes	What is a POMDP, why is this one, what would the alternatives cost?	measured perceptual aliasing with conflicting oracle labels
Policies and oracles	What is a policy, what is the oracle, what exactly is an expert label?	policy spectrum; oracle falsification attempt
Learning paradigms	Is this reinforcement learning, and if not, what is it?	from-scratch Q-learning; first cloned policies
The policy network	What exactly is the architecture, and does every piece earn its place?	generated shape walkthrough; five-way ablation
When cloning breaks	Why does imitation fail under its own mistakes, and what are recovery labels?	corruption sweep; mini three-arm preview
Measuring honestly	What is intention-to-treat, and why budgets, pairing, and freezing?	bias simulations with known ground truth; tamper demonstrations

The rhythm. Each chapter moves from intuition to a precise definition, then to something built and measured, then a common misconception, and finally the bridge to the study.

16 The world

Intuition. Before anything can be learned, there must be a world to act in. Ours is deliberately austere: a small maze of connected rooms, a handful of colored objects, an agent that can turn and walk, and a one-line instruction such as "go to the grey box". Austerity is the point: every later claim about learning, failure, and recovery can be checked by looking at the world directly.

Definition. MiniGrid is a family of procedurally generated grid worlds; BabyAI adds language missions and levels on top of it. The study's frozen task is `BabyAI-GoToObjMazeS4-v0`: each reset seed generates a fresh maze on a 10x10 cell grid, places objects, and issues a mission. The episode succeeds when the agent stands next to the requested object facing it, and ends unsuccessfully after 144 steps (the environment's own limit).

What the agent observes is not the maze. The observation is a symbolic $7 \times 7 \times 3$ integer tensor, an egocentric, occlusion-aware crop of the cells in front of the agent, plus the view direction (0–3) and the mission string. The three channels are lookup indices into fixed vocabularies (object kind, color, door state), not pixels; walls block sight, so much of a typical view is the "unseen" symbol.

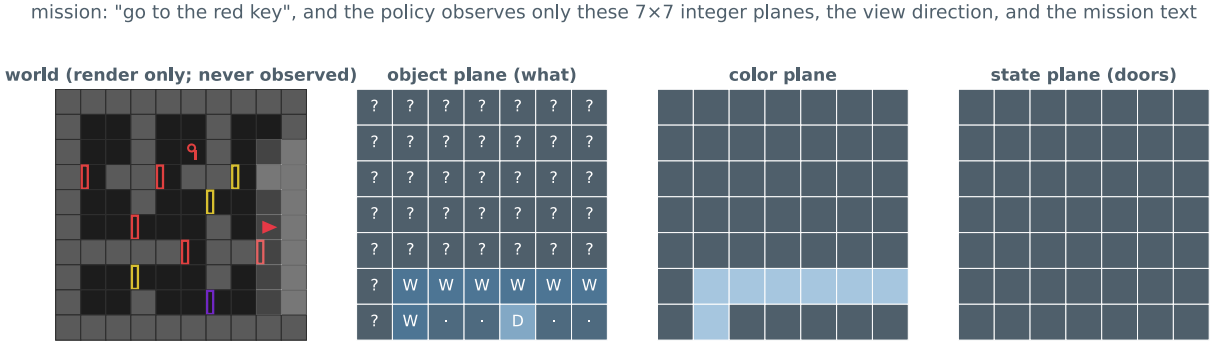

Figure 10: What the policy receives. Left: the world (render only, never observed). Right: the three integer planes of one observation; ? marks cells occluded by walls. The view direction and mission string complete the input.

What the agent can do. The study freezes three actions, left, right, forward, out of MiniGrid's seven. Turning rotates the view in place; `forward` advances one cell when nothing blocks it. Why the other four actions are excluded, and why the set must stay frozen, is derived when corruption operators are introduced (*When cloning breaks*).

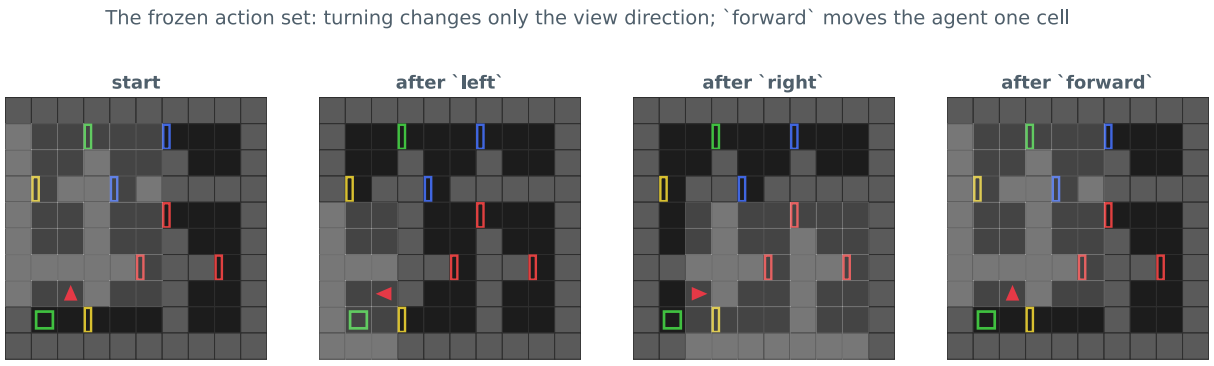

Figure 11: The frozen action set, applied from one start state. The bright cone is the agent's field of view.

Measured census. Over 500 generated worlds, every mission follows one template (`go to a/the <color> <kind>`), spanning 18 distinct strings across six colors and three object kinds, a tiny but real language input: the mission decides which object counts as success. With the contract's `doors_open: true` every door in the census is open; with the environment default most are closed. The full tables (vocabularies, mission distribution, door contrast) are generated in `report/generated/foundations/lab01/`.

Common misconception. "The agent sees the maze." It never does: the world exists in full only inside the simulator. Everything downstream, meaning the need for memory, the value of a privileged teacher, and the meaning of recovery, follows from the gap between the world's state and the agent's observation, which the next chapter formalizes.

Bridge to the study. The study wraps exactly this environment in `grounded_recovery.world.WorldSession`, adding contract checks rather than features: resets require an explicit seed, only frozen actions pass, stepping after termination is an error, and the reset world can be hashed into a scenario identity used by the manifests.

17 Decision processes

Intuition. A decision process is the minimal mathematics of acting over time: where you can be, what you can do, what happens next, and what counts as success. The one distinction that shapes this entire study is whether the agent gets to see where it actually is.

Definition. A **Markov decision process** (MDP) is a tuple (S, A, T, R): states, actions, a transition rule $T : S \times A \to S$ (deterministic here), and a reward R. A **partially observable** MDP adds an observation space Ω and an observation function $O : S \to \Omega$; the agent never receives the state s, only $o = O(s)$. Our task maps onto this tuple exactly: the generated table `pomdp_mapping` pairs each symbol with the code entity that implements it (the full grid and agent pose are the state; the 7×7 crop, view direction, and mission are the observation; reward is sparse and terminal; the horizon is the step limit). All randomness sits in world generation: given the reset seed, dynamics are deterministic.

Aliasing, measured. Partial observability is not an abstract worry; it is countable. Across 300 expert episodes (3928 visited states, collected with the synchronized oracle of *Policies and oracles*), the states collapse into 3746 distinct observations. Of these, 54 observation classes are **aliased**: the same bytes arise from provably different world states, 54 of them from entirely different mazes, and 5 carry **conflicting oracle actions**: the same input, two different correct outputs.

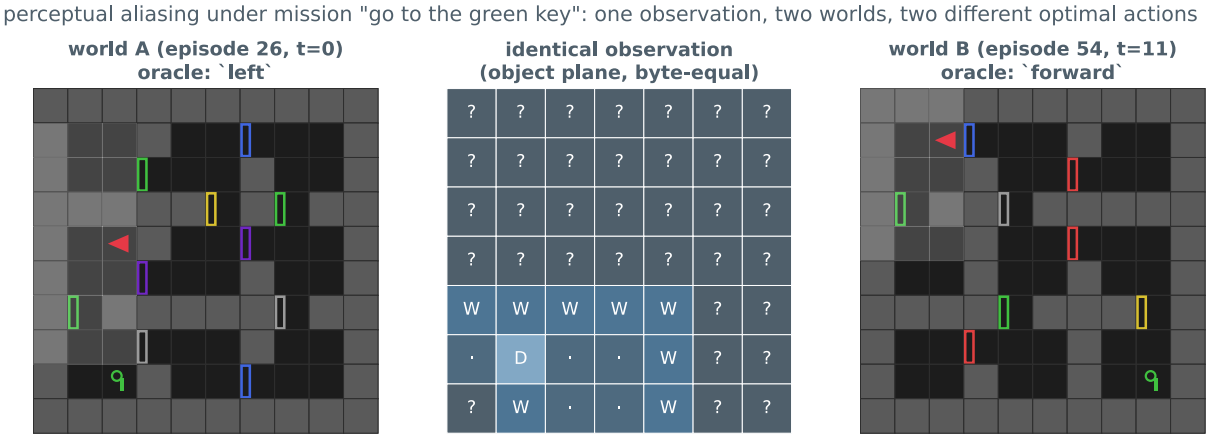

Figure 12: Perceptual aliasing, exhibited. Two different mazes under the same mission produce byte-identical observations while the oracle's optimal actions differ. Selection rule (disclosed): the conflicting cross-world class whose first member appears earliest in rollout order.

Consequence. Any memoryless policy, meaning any function from single observations to actions, must disagree with the oracle on at least 3.0% of visited states on this distribution, no matter how it is trained. Memory is not an architectural taste; it is required for optimality. This bound reappears twice: when the first cloned policies are compared (*Learning paradigms*) and when the architecture's recurrence is ablated (*The policy network*).

Alternatives, and why they are rejected. MiniGrid can hand the policy the entire grid (`FullyObsWrapper`), turning the task into an MDP, but no physical agent observes the world state, and the study is about acting under realistic perception. Frame stacking approximates memory with a fixed window; belief states are exact but require a known world model. The study's choice, learned memory in a recurrent network, is built in *The policy network*.

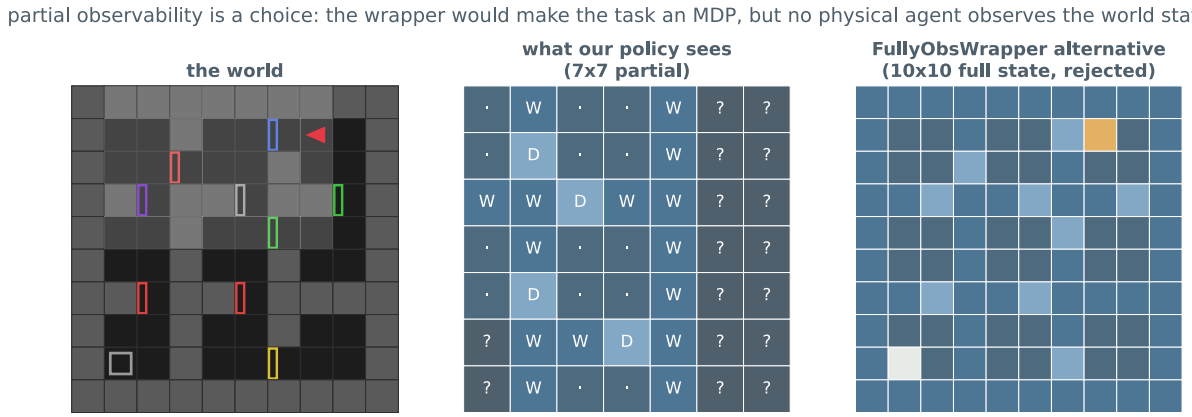

Figure 13: The rejected alternative: full observability would dissolve the problem this study is about.

Common misconception. "Partial observability is a nuisance to engineer around." It is the problem class itself: with the state visible, recovery from a corrupted action would be a lookup; with a partial view, the policy must carry history and **notice** that something went wrong.

Bridge to the study. The study's policies consume exactly the POMDP interface of this chapter, meaning image, direction, mission, plus their own previous executed action, and nothing else. The oracle, by contrast, reads the full state. That asymmetry (privileged teacher, partially observing student) is what makes expert labels informative, and is the subject of the next chapter.

18 Policies and oracles

Intuition. A policy is any rule that turns what the agent sees into what the agent does. Before learning one, it pays to measure how far non-learned rules get, and to meet the teacher whose answers the whole study spends as currency.

The policy spectrum. Three policies on identical scenarios: a uniformly random policy succeeds in 27.3% of episodes, an honest surprise: the maze is small and 144 steps is long, so blind wandering does stumble onto the target, though hopelessly inefficiently. A hand-written right-hand wall follower reaches 54.3%. The scripted oracle solves 100.0% with a mean path of 12.1 steps. The gap between "sometimes, by luck" and "always, directly" is what learning has to close.

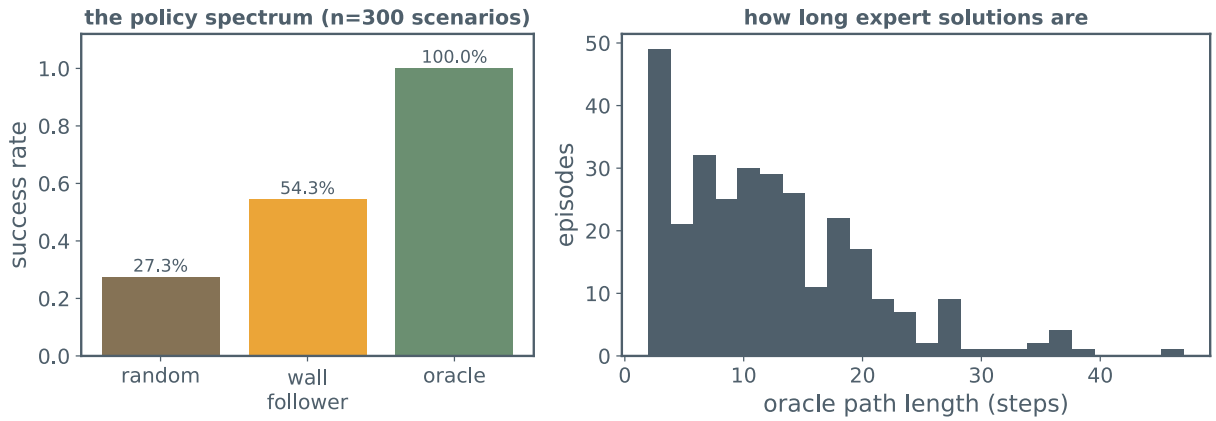

Figure 14: Left: success rates of three non-learned policies on identical scenarios. Right: how long expert solutions are.

What the oracle is. `BabyAIBot` (shipped with MiniGrid) is a scripted planner with privileged access: it reads the full grid, the true POMDP state that *Decision processes* showed the agent never observes, and replans from a subgoal stack at every step. It is not learned and never runs inside the policy. It exists to answer one question at any visited state: *what should be done here?* That answer is an **expert label**. A demonstration is a sequence of such labels along the oracle's own path:

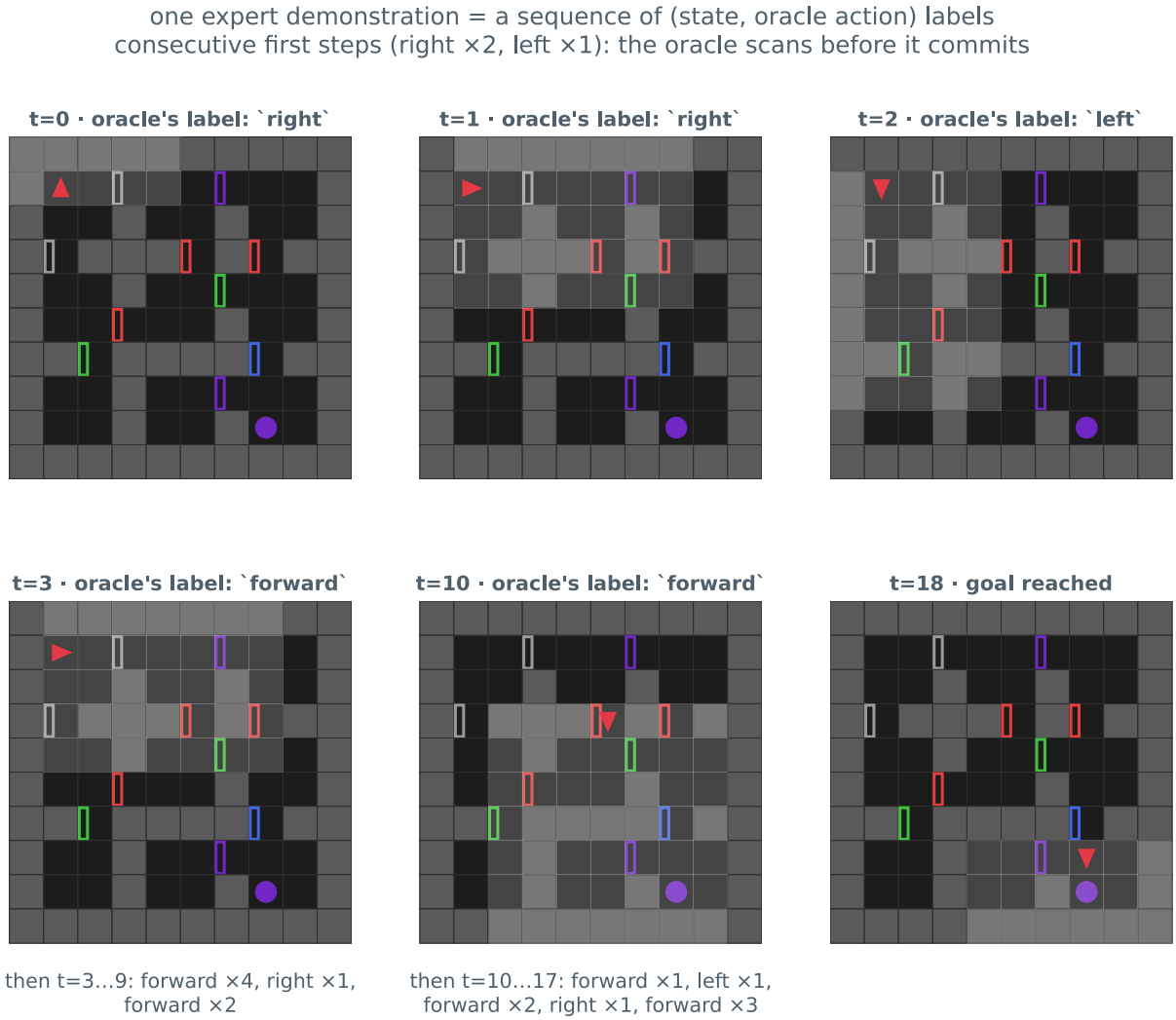

Figure 15: One expert demonstration as the learner consumes it: a sequence of (observation, oracle action) pairs, ending in success.

Recovery competence, and a falsification attempt. Two separate questions, measured on 163 identical scenarios with one forced off-proposal action at a random early step. First, the fact the study rests on: an honestly informed oracle **recovers**, with 100.0% success after the deviation, which is what makes it able to label states the learner reaches instead of the expert (*When cloning breaks*). Second, an attempt to break the bot's bookkeeping three ways: lying about which action was executed (100.0%), never informing it (100.0%), and calling its replanner twice per step (100.0%). Nothing degrades: the bot holds a live reference to the environment and replans from the true world state, and pure navigation barely uses the executed-action bookkeeping (it matters for pickup/drop/toggle subgoals, which the frozen action set excludes).

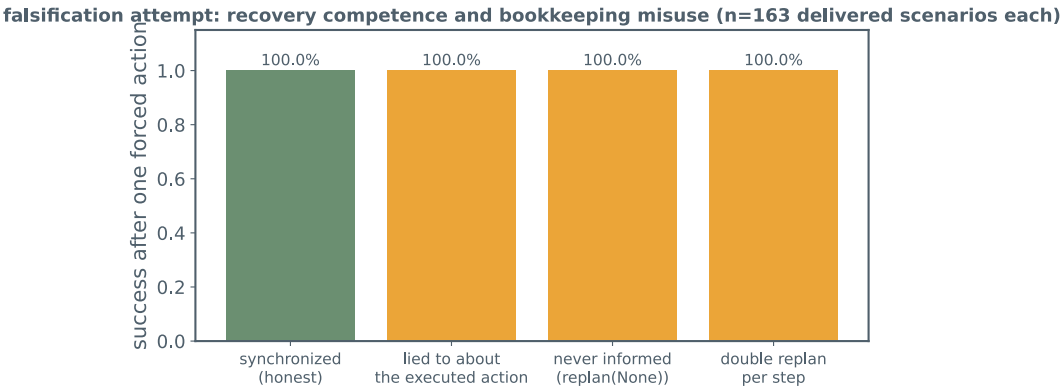

Figure 16: The falsification attempt: recovery competence (left bar) and three misuse protocols. On this movement-only task, none degrades the oracle.

The honest conclusion: on this task the study's synchronization contract ("exactly one replan per executed step, always told the executed action") is not fragility protection, it is **accounting** protection. An oracle query is the unit of supervision the study budgets and ledgers, so the contract is what makes "N labels" a well-defined, auditable quantity.

Common misconception. "The oracle helps the policy at run time." It never does: at evaluation the policy is alone. Oracle answers reach the policy only as training labels, under explicit budget accounting.

Bridge to the study. All of the study's collectors and evaluators drive episodes through one shared loop (`run_synchronized_episode`) implementing the honest protocol above, and the oracle's recovery competence had to pass a preregistered 0.99 preflight gate before any learning experiment ran.

19 Learning paradigms

Intuition. Two ways to learn to act: try things and keep what pays (reinforcement learning), or copy someone who already knows (imitation learning). This study is often mistaken for the former. It is the latter, and the distinction is worth earning with running code rather than asserting.

Definition. Reinforcement learning maximizes expected return through interaction,

$$\max_{\pi} \ \mathbb{E}\left[\sum_{t} R(s_t, a_t)\right],$$

learning from reward alone. Imitation learning fits the policy to labelled expert decisions,

$$\min_{\pi} \ \mathbb{E}[\ell(\pi(o_{1:t}), a_t^*)],$$

supervised learning on (observation history, expert action) pairs. The study is pure imitation: no gradient anywhere depends on reward; reward is read only at evaluation time, to **measure** success.

Real RL, where it belongs. Tabular Q-learning, implemented from scratch, on a tiny fully observable single-room task: the state (agent position and direction, with 32 states ever encountered) is read directly from the simulator, and the greedy policy reaches the goal in 6 steps after a few hundred episodes. RL genuinely works when the state is visible and enumerable and reward is reachable.

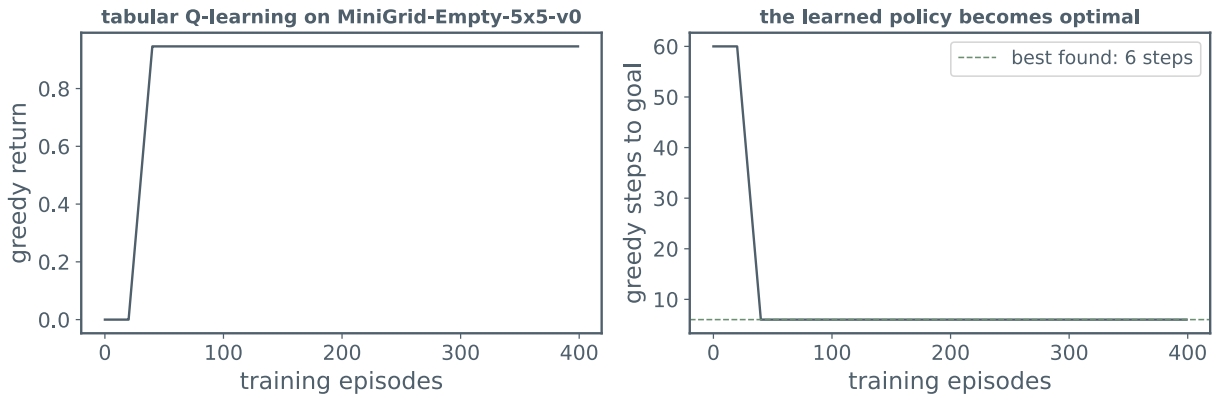

Figure 17: From-scratch tabular Q-learning on a fully observable toy MDP: greedy return (left) and steps to goal (right) during training.

Why that recipe does not carry over. The study task generates a fresh world per seed (no table can enumerate it; *Decision processes* counted thousands of distinct observations in a few hundred episodes), hides the state behind a 7×7 view, conditions success on a mission string, and pays reward only at the end. None of this makes deep RL impossible: random walking already succeeds 27.3% of the time (*Policies and oracles*), so exploration would find reward. The study is not asking an RL question. It asks a supervision-economics question (**which labels help more**, *When cloning breaks*), and behavior cloning is the controlled substrate for answering it.

The first cloned policies. 200 oracle demonstrations (2663 labelled steps) were cloned into two architectures on identical data, identical optimizer, three seeds each: a memoryless policy and the study's recurrent one (all training on cuda).

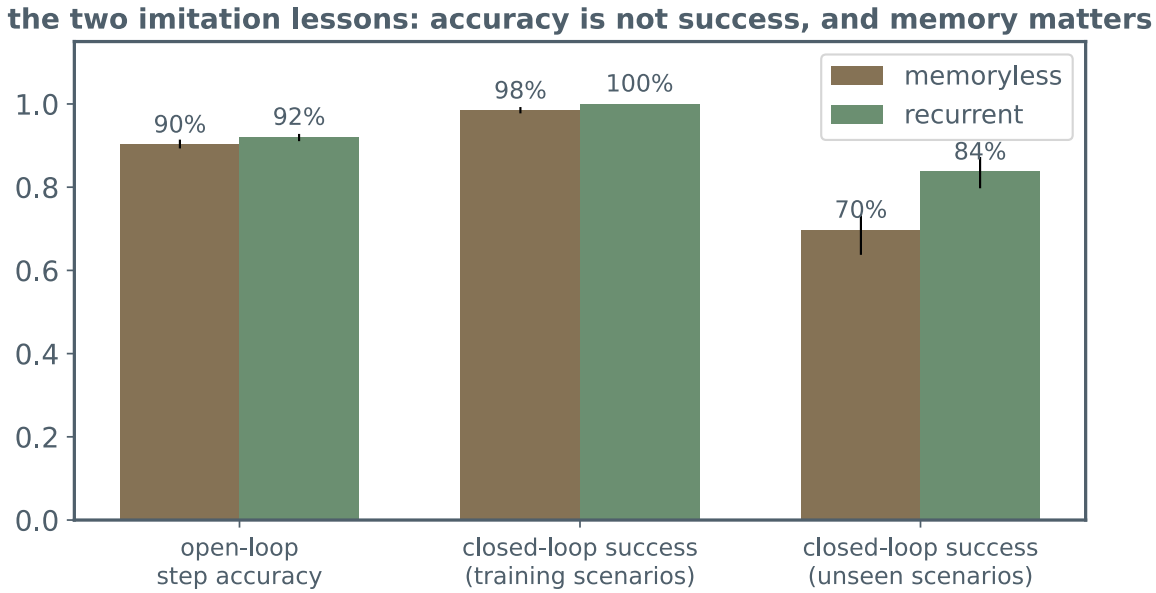

Figure 18: The two imitation lessons. Bars: means over three seeds; whiskers: seed range.

Two lessons, both visible in the figure. (1) **Accuracy is not success**: the memoryless policy matches the oracle on 90.2% of held-out expert steps yet completes only 69.7% of unseen episodes closed-loop, because small per-step errors compound over a rollout, the phenomenon *When cloning breaks* makes central. (2) **Memory matters**: the recurrent policy reaches 92.0% accuracy and 83.7% unseen success, the end-to-end trace of *Decision processes*'s aliasing bound. (A caveat on effect sizes at three seeds is developed in *The policy network*.)

Common misconception. "High step accuracy means a good policy." The memoryless column refutes it: closed-loop evaluation multiplies per-step errors; open-loop accuracy quietly conditions on the expert's own states. This is why the study's primary endpoint is closed-loop task success, never prediction accuracy.

Bridge to the study. The study trains the same recurrent architecture with the same masked cross-entropy idea, but on one-target-per-window items with exact revealed-label budgets instead of whole episodes, a tightening whose reason (*Measuring honestly*) is fairness accounting, not modelling.

20 The policy network

Intuition. The model is small enough to hold in one head, at 179507 parameters, and this chapter makes sure of it three ways: a walkthrough generated from a live instance, a from-scratch reimplementation checked against the original, and an ablation in which the data judges each design choice.

The architecture. Three embedding tables turn the symbolic observation's object/color/state indices into vectors (the observation is symbolic, so embeddings replace the pixel convolutions a visual task would need); two 3×3 convolutions and a linear projection summarize the 7×7 view. A word-embedding GRU encodes the mission once per episode. Direction and previous-executed-action embeddings complete the inputs; a fusion layer mixes everything; a GRU carries memory across steps; a linear head emits three action logits. The full component table, with parameters and live output shapes, is generated by instantiating the real model with forward hooks (`shape_walkthrough`), so the documentation cannot drift from the code. A from-scratch reimplementation (`gr_foundations.models.LabPolicy`) matches the study model parameter for parameter; the parity is asserted at runtime and in tests.

component	parameters	output shape (B=2, T=5)
object_embedding	176	2x5x7x7x16
color_embedding	96	2x5x7x7x16
state_embedding	64	2x5x7x7x16
observation_conv	23104	10x32x3x3
observation_projection	18496	10x64
word_embedding	672	2x6x32
language_gru	18816	2x6x64
direction_embedding	32	2x5x8
action_embedding	32	2x5x8
fusion	18560	2x5x128
policy_gru	99072	2x5x128
head	387	2x5x3
total	179507	

One component removed at a time. Five variants (the full model, no policy GRU, no mission input, no previous-action input, and a bag-of-words mission encoder) trained on identical demonstrations with an identical optimizer, three seeds each, judged by closed-loop success on unseen scenarios:

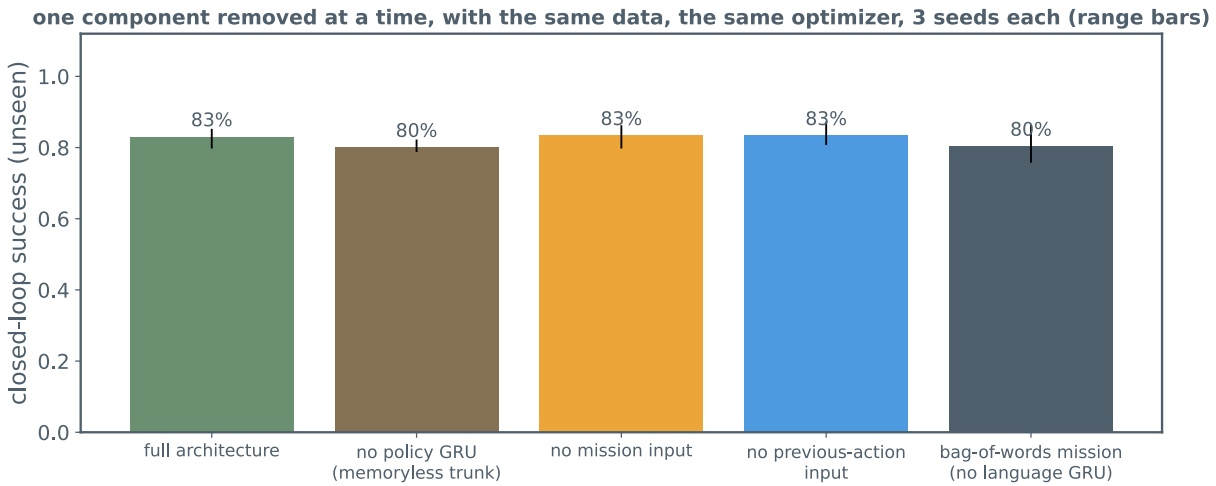

Figure 19: Matched ablation (three seeds; whiskers span the seed range).

Reading the ablation honestly. At this training scale, clean-condition success is a blunt instrument: all five variants land within a few points, with heavily overlapping three-seed ranges, and the same architecture trained under this chapter and under *Learning paradigms* does not land in the same place twice. Three-seed comparisons wobble, which is precisely why the study runs six paired replicates and reports an interval (*Measuring honestly*). Two results deserve their own sentences. Removing the mission costs nothing **here**: with one distractor and a 144-step limit, a mission-blind policy that tours objects still ends on the right one, because the endpoint tolerates detours, so language conditioning earns its keep only at tighter horizons or richer scenes (an interpretation, marked as such). And the bag-of-words mission encoder matches the GRU at this five-word grammar.

Why the architecture is still the right one. The components are justified by their roles in the study, not by clean-run ablation wins. Memory: *Decision processes* proved no memoryless policy can match the oracle on aliased states, a structural argument independent of seed noise. Previous-action input: nearly free in clean conditions, but it is the only channel through which an externally corrupted execution becomes visible to the policy; its purpose exists only under *When cloning breaks*'s corruptions. Mission conditioning defines the task. And a recurrent core over a Transformer: the dataset is a few thousand labelled steps, inference is streamed one observation at a time with $O(1)$ state, and the model stays small enough to audit by hand, which this repository treats as a feature.

Common misconception. "The ablation with the best number is the right architecture." At small n, ablation deltas smaller than seed noise decide nothing; design arguments and preregistered endpoints have to carry the weight the noise cannot.

Bridge to the study. `grounded_recovery.model.RecoveryPolicy` is exactly this architecture; the study adds nothing at model level. Its checkpoints additionally freeze the vocabulary, action names, and all RNG streams so training can be resumed or cloned bit-exactly, the cloning that lets three arms start from the **same** base checkpoint in *When cloning breaks* and in the study.

21 When cloning breaks

Intuition. Behavior cloning is trained on the expert's states but deployed on its own. The first wrong action moves the policy somewhere the demonstrations never covered; whatever it does there is unsupervised, and errors feed on themselves. This chapter measures that spiral, formalizes the corruption operators used to trigger it on demand, and previews the study's remedy comparison at small scale.

The failure mode, measured. The compounding-error argument (Ross and Bagnell's classical analysis: per-step error ε can cost on the order of εT^2 over a horizon T, precisely because mistakes change the state distribution) becomes concrete with one corrupted executed action:

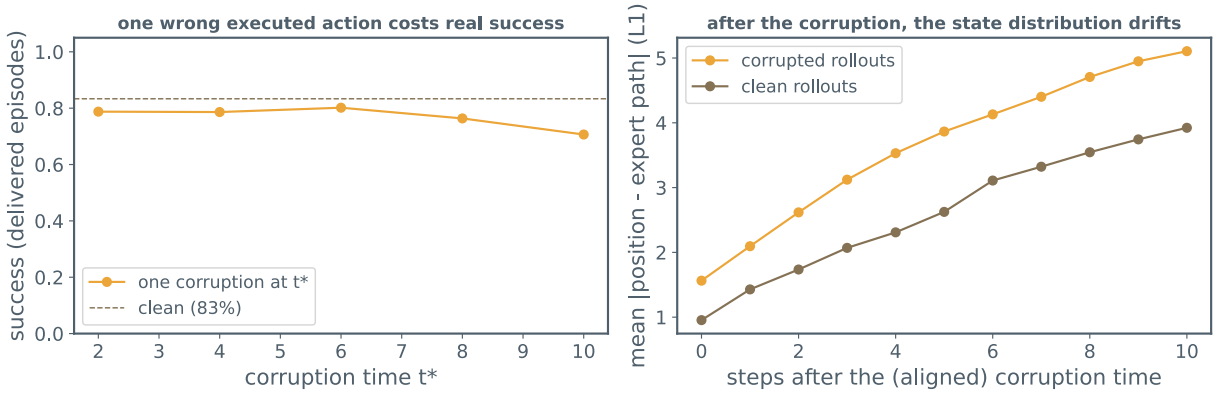

Figure 20: Left: success of the base policy when exactly one executed action is corrupted at time t^* (delivered episodes), against its clean rate. Right: mean distance to the expert path after the (aligned) corruption time, since clean rollouts drift apart from the expert's route slowly (valid alternative paths diverge positionally too); corrupted rollouts drift roughly twice as far.

Corruption operators, and why the action set is frozen. A corruption must change every action it touches, otherwise some corruptions would be no-ops; an operator is therefore a **derangement** (fixed-point-free permutation) of the action set. For three actions exactly two derangements exist: the two 3-cycles, each the other's inverse (`derangements` table): the study uses one during collection and holds out the other as the unseen operator. Changing the action set would change this entire operator family; enlarging it would also break oracle support, since with closed doors the bot emits `toggle`, outside the frozen set, which is exactly how the study's `doors_open: true` parameterization was discovered during piloting (*The world*). The action set is a frozen contract field, not a tuning knob.

mapping	effect	role in the study
(1, 2, 0)	left->right, right->forward, forward->left	rot_plus (collection)
(2, 0, 1)	left->forward, right->left, forward->right	rot_minus (unseen)

Two remedies, one budget. Both arms start from the **same** base weights (trained on 2101 labels) and receive exactly 400 additional oracle labels each. The **extra** arm spends them on fresh nominal demonstrations. The **recovery** arm spends them where the learner actually goes: the base policy rolls out, one action is corrupted, and the oracle, kept synchronized along a trajectory it did not choose, labels the next 8 states the policy visits. Optimization is matched (same update count, same fixed base/new batch mix). One fairness leak is deliberately left open and **measured**: an extra-demo batch carries 54.2 new labels per update against 29.7 for recovery batches, because demonstrations are label-dense while recovery windows are sparse. The study closes exactly this leak with one-target-per-window training items (*Measuring honestly*).

What the mini-study shows.

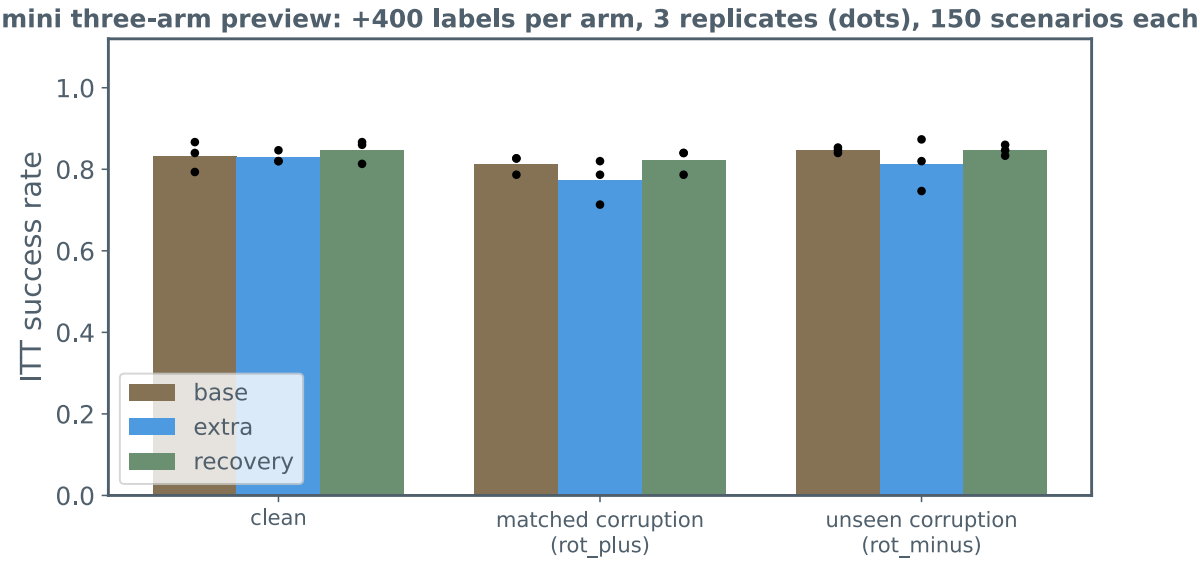

Figure 21: Intention-to-treat success by evaluation slice (three replicates as dots). Exploratory preview at small scale, not confirmatory evidence.

Across three replicates the recovery-minus-extra difference on the unseen-operator slice spans -1.3% to +10.0% (mean +3.3%), with corruption delivery at 88.3%. Two honest observations. First, one corruption dents this base policy by only +2.0% on the matched ITT slice, and a strong base leaves little headroom for either remedy, one reason the study's frozen protocol checks perturbed competence and evaluates a far larger panel. Second, three replicates cannot even fix the **sign** of the difference, the small-n lesson the next chapter turns into design requirements. The confirmatory answer, over 6 replicates and 536 eligible scenarios under the frozen protocol, is Part I's +14.30 pp with a 95% interval of [+10.02, +18.59] pp (*Results*).

Common misconception. "Corruptions are adversarial attacks." They are a **probe**: a controlled, disclosed way to place the policy off its training distribution exactly once, so that recovery behavior becomes measurable and comparable. The unseen operator exists so that the measurement cannot reward memorizing the probe itself.

Bridge to the study. Everything here mirrors `grounded_recovery.experiment` at reduced scale; the study adds bit-exact arm cloning, hash-chained exposure ledgers with a fairness audit, exact window-level target accounting, preregistered corruption time sets, and the one-opening evaluation discipline of the next chapter.

22 Measuring honestly

Intuition. By this point the question is sharp: under equal additional label budgets, do recovery labels beat extra demonstrations? What remains is everything that makes an answer **trustworthy**, and every piece of that discipline exists because a specific, demonstrable bias would otherwise creep in. This chapter demonstrates the biases.

Intention-to-treat, defined. Analyze by what was **scheduled**, not by what happened. Every perturbed evaluation episode has a scheduled corruption time; sometimes the episode ends first and the corruption is never delivered. Delivery depends on the policy's own behavior, so it is a post-treatment variable, and conditioning on it compares filtered, non-comparable subsets.

The bias, with known ground truth. In a simulation built so that the true ITT effect is +0.070 by construction (and the better arm, being more efficient, ends more episodes before late corruptions arrive), the ITT estimator's bias is -0.0010 while the per-protocol estimator, using only delivered episodes, is off by -0.0355: a bias of the same order as the effects this field measures, produced by nothing but a filter that looks innocuous.

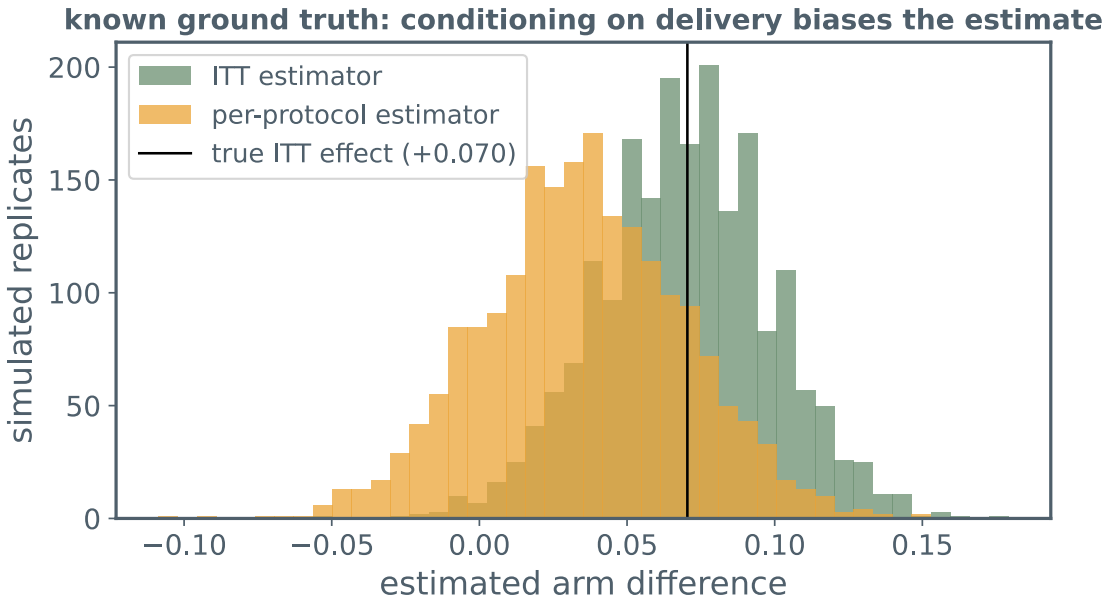

Figure 22: 2,000 simulated replicates against a known truth: the ITT estimator centers on the estimand; conditioning on delivery does not.

On the real rows of *When cloning breaks* the two estimates happen to agree (+3.3% ITT versus +3.8% delivered-only) because delivery is nearly universal there. The design still preregisters ITT: delivery **could** differ between arms, and nothing in the data would warn you.

Budget matching. The study's currency is revealed oracle labels, and the match was deliberately broken to see what happens: the recovery arm reran with twice the budget and landed at 83.3% unseen success against 84.7% matched. The feared inflation did **not** materialize This base sits near its headroom ceiling (*When cloning breaks*), so doubled supervision bought nothing measurable in a single replicate. That is the deeper point: an unmatched design attributes to the **method** whatever the extra **budget** did or did not do, and a small unfrozen run cannot even tell you which way the confound cuts. Matching is a design necessity for attribution, not an empirical convenience.

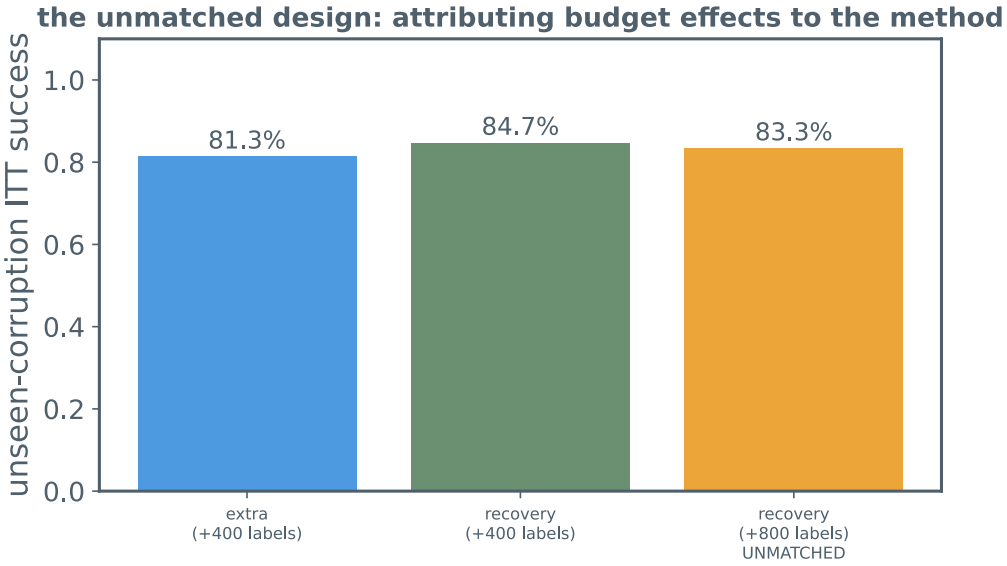

Figure 23: The unmatched design, tried on purpose. Whatever the outcome, the comparison stops measuring the method.

Why paired replicates. Every pipeline bundle re-rolls collection and training, shifting both arms together; analyzing per-bundle **differences** cancels the shared noise. In simulation at six bundles, the paired analysis detects a +0.05 gap with 100% power against 82% for the unpaired analysis of the same data:

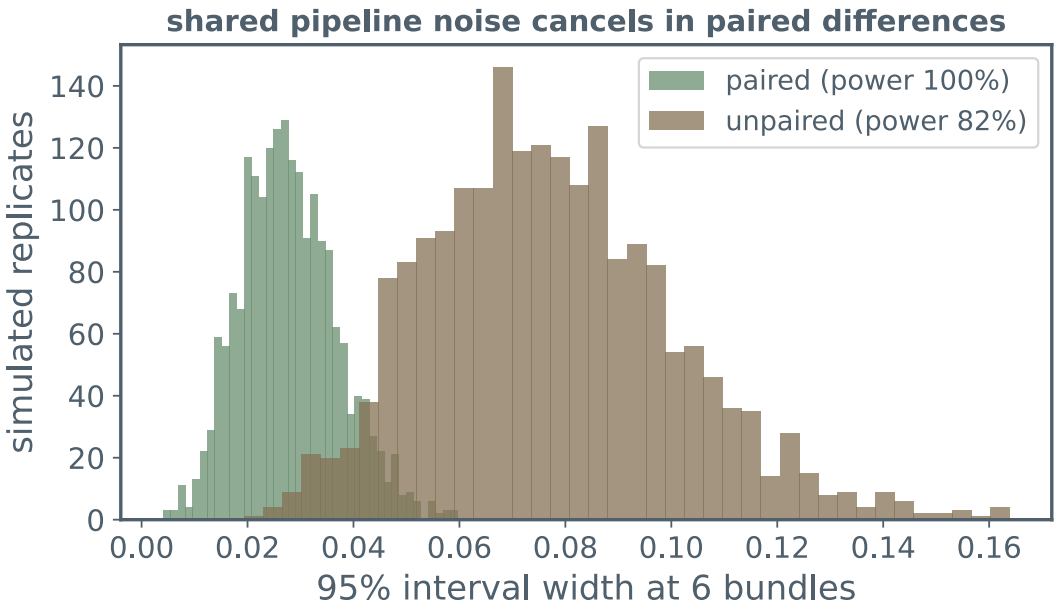

Figure 24: Interval widths over 2,000 simulated replicates: pairing turns six bundles into a usable instrument.

Freezing, and what it is made of. A frozen protocol is a set of mechanical commitments, not a promise. Two of them in miniature: flipping a single contract field changes the contract's canonical fingerprint completely, so a quiet edit cannot survive comparison against the recorded one; and editing one row of a five-row hash-chained ledger is caught at exactly row 2 when the chain is recomputed. The full mechanism table:

mechanism	**failure it prevents**	**where**
PILOT_TO_FREEZE placeholders	running confirmatory stages with unresolved design choices	config.py refuses FROZEN status while placeholders remain
contract hash (canonical JSON, SHA-256)	silent post-hoc edits to the design	every artifact stamps the hash; one flipped field changes it
scenario manifests with identity hashes	evaluation scenarios drifting between runs	data.py manifests + disjointness audit
hash-chained ledgers + independent recount	quiet insertion, deletion, or edit of collected data	integrity.py LedgerWriter / recount_dataset
single receipted test opening	rerunning the confirmatory evaluation until it looks good	experiment.py opening receipt; write-once results
preregistered claim decision rule	choosing the interpretation after seeing the numbers	frozen text mapping interval position to support/adverse/rule-out/inconclusive

Common misconception. "Preregistration is bureaucracy." Each mechanism above is a bias with a name, blocked mechanically: selective delivery (ITT), unequal generosity (budget match), shared-noise dilution (pairing), silent redesign (contract hash), quiet data edits (chained ledgers), rerun-until-pretty (single receipted opening), and post-hoc interpretation (the preregistered claim decision rule).

Bridge to the study. Part II's endpoint is the eligible unseen one-corruption ITT success difference, analyzed as a paired t interval across six bundles against a 0.05 smallest effect of interest, under a preregistered claim decision rule, on data whose ledgers recount cleanly and whose test set was opened exactly once against a

receipt. Every one of those words was one of this chapter's demonstrations, done, in Part I, at full scale. You now have everything needed to audit it.