FORKPRM: LEARNING EMBODIED PROCESS VERIFIERS FROM EXECUTABLE COUNTERFACTUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process supervision improves language-model reasoning, but step-level correctness labels are hard to obtain for embodied agents because an action's quality depends on its downstream environmental consequences. We introduce ForkPRM, which turns that dependence into a supervision signal: during training, candidate actions are executed from the same observed navigation history and each resumes the same frozen continuation policy, so the measured utilities of the resulting branches give baseline-relative process labels without human annotation. These labels train a candidate-set verifier that scores candidates at application time and selectively overrides or defers to the policy action without executing additional environment forks. On R2R vision-and-language navigation, shared-prefix supervision raises ordinary pairwise ranking over unpaired outcome regression in the same five-seed supervision panel and label budget (+2.4 pp, positive in all five seeds; +7.6 pp over a cross-state permutation control), but these ranking gains do not extend to utility-weighted accuracy, rescue recall, or harm rate. A one-shot evaluation on scenes reserved from the verifier's splits—drawn, like the policy's own training, from the R2R training-environment pool—yields a small branch-selection gain whose confidence interval includes zero against a much larger budget-matched oracle; ranking accuracy is therefore not yet policy utility.

1 INTRODUCTION

Learning agents that act in environments currently receive most of their learning signal at the trajectory level. An outcome reward tells the agent whether the whole episode succeeded, but not *which* decision was responsible; a value function trained on such returns must disentangle action quality from environment stochasticity without any direct observation of step-level correctness. Process supervision—grading individual steps—has been transformative for language-model reasoning (Lightman et al., 2024; Uesato et al., 2022), but there the labels come from human annotation of intermediate solutions or strong teacher models. For an embodied agent, no such annotator exists: whether turning left at a hallway junction is "correct" depends on what the environment reveals afterwards, not on the action's surface form.

This paper starts from an observation about simulators rather than about annotators. *An embodied simulator can execute multiple candidate actions from exactly the same state.* During training, we can branch the environment at a decision point, force one candidate action, continue with the same frozen policy under deterministic replay, and observe the true downstream consequence of that single intervention. The difference between two such branches is a *counterfactual, baseline-relative process label*: it measures what the candidate actually caused, relative to what the policy would have done, under a matched history. No human or teacher judgment is involved; the environment is the annotator.

We ask: *how can an embodied agent learn a process verifier when step-level correctness is not directly observable, but interventions can be executed during training?* Our answer is ForkPRM (*Fork* process reward modeling): at training time, fork the policy action together with up to $K-1$ ranked alternatives (K arms total) at three predefined early/middle/late decision indices of a frozen navigation policy and roll out each branch; at application time, the learned verifier only scores

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.

the candidate set and may override the policy's choice under conservative, calibrated conditions—*branches are rolled out only during training; application performs no additional forks*.

ForkPRM makes three contributions. First, we formalize *executable counterfactual supervision*: paired same-prefix action interventions with a frozen continuation policy, whose shared-prefix structure reduces label variance under an additive decomposition of utility into state and action components, and empirically improves ordinary candidate ranking by +2.4 pp (0.711 vs. 0.687, positive in all five seeds) over unpaired outcome training in the same supervision panel and label budget; a cross-state permutation control that destroys the state–input–outcome correspondence widens the gap to +7.6 pp and bounds the contribution of true correspondence from above. Second, we distill these labels into a candidate-set process verifier and evaluate it as a controlled package against unpaired outcome supervision and against ablated variants. Third, we evaluate under a *pre-registered, one-shot confirmation protocol*: all design and threshold decisions are frozen on development splits, and the final held-out confirmation set is executed exactly once.

The contribution is the supervision construction and its controlled evaluation: executable forks improve ordinary consequence ranking, and the same evaluation measures how far that ranking is from policy utility (Sections 5.1 and 5.3).

2 Related Work

Process supervision for reasoning. Step-level reward models improve final-answer supervision in mathematical reasoning (Lightman et al., 2024; Uesato et al., 2022); Math-Shepherd shows such process labels can be estimated automatically from rollouts rather than human annotation (Wang et al., 2024); and prover–verifier games strengthen verifiers with adversarially difficult solutions (Kirchner et al., 2024). In reasoning, such labels come from human annotation, teacher models, or automatic continuation statistics; in all cases correctness is a property of the solution text. ForkPRM's difference is the supervision source: process labels are measured consequences of *shared-state physical action interventions* executed by an embodied simulator under a fixed continuation policy, and the harmful alternatives that a prover-style scheme would mine arise here directly as candidates that look plausible to the policy yet measurably harm the trajectory.

Progress estimators and verification in VLN. Navigation has a long line of learned progress monitors and candidate scorers, from self-monitoring agents (Ma et al., 2019) and history-aware progress estimators (Qiao et al., 2022) to dual-scale cross-modal graph planning (Chen et al., 2022) and hierarchical topological planners (An et al., 2024). Existing monitors are typically trained from sub-goal annotations, heuristics, or auxiliary objectives; some approaches additionally consult short rollouts at decision time. ForkPRM differs in the supervision itself: labels are counterfactual differences between *executed* candidate actions from an identical state under a frozen policy, requiring no annotation and no additional counterfactual branch rollouts at application time.

Rollout-based action values and classification-based policy improvement. Estimating per-action values by rolling out candidate actions and improving a policy by classification or advantage-weighted regression has a long history, including classification-based policy iteration and its rollout action-value estimates (Lazaric et al., 2016). ForkPRM adapts this machinery to embodied supervision: hash-verified shared-history interventions executed in the simulator under a fixed continuation policy, distilled into a candidate-set verifier, with the ranking–utility gap characterized under a one-shot protocol.

Paired estimation and variance reduction. Comparing interventions under shared randomization is a classical variance-reduction technique; our shared deterministic prefix is its simulator-native analogue. We show the effect transfers to *learning*: training on same-state paired labels beats training on unpaired outcomes at matched data size (Section 5.1).

Selective policy correction. At application time the verifier overrides a frozen policy only when its margin and predicted-success gap exceed calibrated thresholds, otherwise deferring to the policy. The conservative design is motivated empirically: always taking the second-ranked candidate costs -13.23 SPL in our evaluation (Section 5.3).

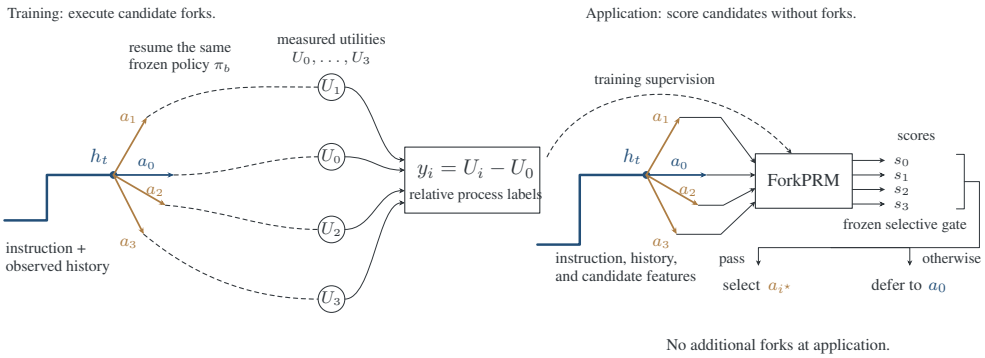


Figure 1: **Overview of ForkPRM.** During training, candidate arms share the same observed history; each arm executes one candidate action and then resumes the same frozen continuation policy, producing measured utilities that yield relative process labels. At application time, ForkPRM scores the candidate set and selects an alternative or defers to the policy action a_0 (blue) without executing additional environment forks.

3 METHOD

3.1 EXECUTABLE COUNTERFACTUAL SUPERVISION

Let π_b be a frozen navigation policy, h_t its decision history (instruction, topological map, observations), and $\mathcal{A}_t = \{a_0, \ldots, a_{K-1}\}$ the forked candidate set at step t: a_0 is the policy's own action and $a_1, \ldots, a_{K-1}$ its top-ranked legal alternatives, for K arms in total. We use $K{=}4$ for the final ForkPRM training and confirmation evaluation; the pilot scaling analysis additionally evaluates $K \in \{2, 3, 4\}$. At a designated *fork step* t, we replay the episode from reset under a deterministic seed so that h_t is bit-identical across branches (verified by state hashes), force $a_t = a_i$, and let π_b continue to termination. The terminal utility $U(\tau^{a_i})$ (SPL in our experiments) of each branch defines the counterfactual process label

$$y_t^{i,0} \;=\; U(\tau \mid \mathrm{do}(a_t{=}a_i), \pi_b) \;-\; U(\tau \mid \mathrm{do}(a_t{=}a_0), \pi_b)\,, \tag{1}$$

where a_0 is the policy's own choice. The label is *causal* (a single intervention, everything else matched), *baseline-relative* (it grades deviations from the policy, not absolute action value), and *free of annotator judgment*. Because all branches share h_t, if utility decomposes additively into a state component and an action component, the paired difference $y^{i,0}$ cancels the state component. Empirically, the paired differences on our data have $1/1.78\times$ the variance of differences computed across states (an empirical variance ratio, not an unconditional guarantee).

We place forks at three predefined early/middle/late decision indices along each trajectory and roll out the policy action together with up to $K{-}1$ ranked alternatives (K arms total per fork), so every forked state yields a ranked candidate set with measured consequences. Forks are executed during training only; the deployed verifier performs no additional counterfactual forks.

3.2 CANDIDATE-EQUIVARIANT PROCESS VERIFIER

Each candidate a_i is described by (i) the frozen policy's cross-modal node embedding after graph fusion, (ii) the *pre-fusion visual node representation* of the corresponding map node, (iii) geometric relation features, policy logit and probability, and a stop indicator. The state contributes the stop-node embedding, a mean cross-modal graph embedding, and the mean instruction embedding. A shared trunk f maps each candidate row to $h_i \in \mathbb{R}^d$; an invariant context $c = \mathrm{mean}_{i \in \mathrm{valid}} h_i$ summarizes the set; and heads consume $[h_i,\, c,\, h_i - h_0]$, where $h_i - h_0$ makes the baseline-relative structure explicit. Equivariance is defined with respect to permutations of the alternative candidates while preserving the identity of the designated policy-action baseline a_0. With that convention the scorer is permutation-equivariant over the candidate set and yields, per candidate, a process score s_i, a success logit, and rescue/harm logits (whether taking a_i converts a policy failure into a success or vice versa), plus an auxiliary delta regressor.

3.3 OUTCOME-STRUCTURED OBJECTIVE

Training combines a listwise ranking loss over realized consequences with outcome-structured supervision and harm-aware weighting:

$$\mathcal{L} = \mathcal{L}_{\text{list}} + \lambda_{\text{out}}\,\mathcal{L}_{\text{out}} + \lambda_{\text{aux}}\,\mathcal{L}_{\text{rescue/harm}} + \lambda_{\delta}\,\mathcal{L}_{\delta}, \tag{2}$$

where $\mathcal{L}_{\text{list}}$ is cross-entropy of $\mathrm{softmax}(s)$ at the candidates with the best realized utility, $\mathcal{L}_{\text{out}}$ a per-candidate success BCE, and $\mathcal{L}_{\text{rescue/harm}}$ the two structured BCE heads. Every candidate term is weighted by $w_i = 1 + \lambda\,\mathrm{ReLU}(-y^{i,0})$: candidates whose measured label *harms* the trajectory relative to the policy receive up-weighted gradients. The weighting depends only on measured label harm.

3.4 CONSERVATIVE SELECTIVE OVERRIDE

At deployment the verifier performs no additional counterfactual forks. Given the candidate set, we compute $i^{\star} = \arg\max_i s_i$ and override the policy only if the score margin and the predicted-success gap over the policy action both exceed thresholds calibrated on a development split; otherwise the policy action stands. The operating point is chosen to be conservative: the verifier triggers on 9.5% of eligible confirmation states, keeping the policy everywhere else.

4 EXPERIMENTAL PROTOCOL

Setup. We use R2R vision-and-language navigation (Anderson et al., 2018) in the continuous environments of VLN-CE (Krantz et al., 2020), simulated with Habitat (Savva et al., 2019; Szot et al., 2021) on Matterport3D scenes (Chang et al., 2017), with the frozen ETPNav policy (An et al., 2024) (two checkpoints: the released `iter12000` and a weaker `iter4000`). Utility is SPL; success rate serves for the rescue/harm structure. At each of three fork positions per trajectory (decision indices 2/5/9), the policy action plus up to three ranked alternatives are rolled out to termination under deterministic replay (four arms total per fork); branch pairing is verified by hash-identical prefixes, and every arm's trace is audited for a single legal override with the policy resuming afterwards.

Data discipline. All routes are R2R training-split episodes, freshly selected and scene-isolated across roles: *train* scenes for supervision; a *pilot* set of eight never-touched scenes for the initial study; *calibration* scenes (unused in any prior experiment) for threshold and ensemble freezing; and a locked *confirmation* set of 12 scenes / 192 routes whose selection was frozen on September 18, before any ForkPRM experiment had been run, and executed exactly once on September 22. All splits, including confirmation, draw from the R2R training-environment pool (environments seen by the policy during its own imitation training); generalization to the standard val_unseen environments is not evaluated. The pilot evaluation of a simpler ForkPRM-MLP variant and all exploratory analyses preceded v2 development.

Pre-registration. Before touching the confirmation set we froze five submission criteria (branch-selection gain $\geq$ +1.0 SPL with scene-bootstrap CI excluding zero; budget-matched oracle capture $\geq$ 25%; a positive aggregate estimate in $\geq$ 10 of 12 leave-one-scene-out recomputations (a dependent descriptive sensitivity check, since the recomputations share scenes); non-negative transfer to the second checkpoint; improvement over ForkPRM-MLP $\geq$ 0.5 SPL). The one-shot confirmation is reported in Section 5.3. Statistics use a scene-level cluster bootstrap with multiplicity; the main estimand is episode-weighted ΔSPL, with a scene-balanced variant agreeing to within 0.00.

5 RESULTS

5.1 RQ1: DOES EXECUTABLE PAIRING IMPROVE LEARNABILITY?

Table 1 compares verifiers trained with different supervision sources, evaluated on calibration scenes never used for training (five seeds, mean$\pm$sd). The primary comparison is against unpaired outcome regression in the same supervision panel and label budget: shared-prefix training reaches 0.711 versus 0.687 (+2.4 pp; per-seed mean +2.3 pp, sd 0.9 pp, range [+1.6, +3.7] pp, positive in all

Table 1: **Supervision quality** on calibration scenes; all cells are mean±SD over the five independent training seeds (per-seed models, not the frozen ensemble). Utility-weighted accuracy weights each pair by $|\Delta\text{SPL}|$. Rescue recall: among states where the policy fails but some candidate rescues, the fraction where the verifier's top pick is a rescuer. Harm FPR: among overrides of successful policy actions, the fraction that break them.

Supervision	Pairwise acc.	Utility-wt.	Rescue rec.	Harm FPR
Unpaired outcome regression	0.687±0.019	0.674±0.055	0.23±0.14	0.18±0.06
Baseline-relative regression	0.698±0.017	0.684±0.042	0.22±0.11	0.14±0.06
Cross-state permutation control	0.635±0.049	0.622±0.077	0.20±0.12	0.19±0.08
Shared-prefix forks (no visual)	0.711±0.011	0.660±0.051	0.21±0.13	0.24±0.10
ForkPRM (full)	0.686±0.047	0.639±0.086	0.19±0.10	0.25±0.11

Table 2: **Verifier ablations** on calibration scenes (five seeds, mean±sd). All component ablations fall within seed noise of the full model; deployment differences between the full package and the unstructured baseline are reported in Table 3.

Variant	Pairwise	Utility-wt.
No visual candidate repr.	0.711±0.011	0.660
No equivariant structure	0.688±0.047	0.633
No harm-aware weighting	0.693±0.046	0.643
ForkPRM (full)	0.686±0.047	0.639

five seeds), with baseline-relative regression in between (0.698). As a *negative control*, permuting candidate–outcome assignments across states ($\pi : (x_s, y_s) \mapsto (x_s, y_{\pi(s)})$ within equal-cardinality groups) destroys the state–input–outcome correspondence while preserving label marginals; against this control the gap widens to +7.6 pp (five-seed mean $\pm$1.4 pp, per-seed range [+5.6, +9.1] pp). The control bounds the contribution of true correspondence from above; it is not a competitive baseline. The variance analysis of Section 3.1 is consistent with the mechanism: under an additive decomposition of utility into state and action components, shared prefixes cancel the state component; on our labels the paired difference has an empirical variance ratio of $1.78\times$ relative to unpaired arms. The boundary of this effect must be stated plainly: the pairwise gain shrinks for utility-weighted accuracy, vanishes for rescue recall, and the harm false-positive rate is no better than the control (Table 1); pairing improves ranking, not utility-aligned selection. A pilot-scale replication with a smaller architecture found the same direction (0.758 paired vs. 0.671 shuffled on separate held-out scenes). Paired labels do not guarantee transfer—they make the learning problem better posed.

5.2 RQ2: Do candidate-conditioned features and equivariance matter?

Table 2 ablates the verifier's components on calibration ranking. Single-component ablations fall within seed variability ($\pm$0.05 pairwise accuracy), and the no-visual variant is nominally the *best* ranker. We therefore cannot attribute the package-level confirmation behavior to the visual candidate representations, the equivariant structure, or harm-aware weighting individually. The only clean, controlled component effect in this work is the shared-prefix pairing of Section 5.1. What the ablations leave open is branch-selection behavior: on confirmation, the full package is the variant whose frozen gate remains positive under the leave-one-scene-out sensitivity check (Section 5.3) while the unstructured ForkPRM-MLP does not—a package-level difference we can demonstrate but not decompose with these ablations.

5.3 RQ3: Selective override under one-shot confirmation

Table 3 reports the branch-selection evaluation on the confirmation set. This is an *offline logged-prefix* estimate: each decision point is scored on the frozen policy's recorded history, the realized outcome is read from the corresponding fully executed single-override branch, and the headline +0.10 SPL is the per-fork, per-episode average over the three decision indices (**??**); corrections at different forks do not compound, so closed-loop improvement is not measured. The full model gains +0.10 SPL (95% CI [-0.84, +1.23], including zero) and the unstructured ForkPRM-MLP gains -0.22

Table 3: **One-shot branch-selection evaluation** (12 scenes / 192 routes, frozen artifacts; offline logged-prefix, per-fork single override, no compounding across forks). Net ΔSPL is episode-weighted; CI is a scene cluster bootstrap. "Oracle (budget-matched)" overrides the best candidate on the same *number* of interventions as the verifier triggers, with the states chosen by realized oracle gain.

Model	Net ΔSPL	95% CI	Triggers	LOO	Oracle (budget-m.)	Capture
ForkPRM-MLP	-0.22	[-0.87, +0.35]	33	0/12	+4.29	-5.2%
ForkPRM	+0.10	[-0.84, +1.23]	39	12/12	+4.85	2.0%
ForkPRM → ckpt4000	-0.05	[-0.81, +0.66]	37	0/12	+5.79	-0.8%

SPL; the episode-paired difference between the two models is +0.32 SPL (95% CI [-0.61, +1.41]). A leave-one-scene-out sensitivity check (dependent and descriptive: the 12 recomputations share 11 scenes each) separates the two models: the aggregate estimate stays positive in 12/12 recomputations for ForkPRM versus 0/12 for ForkPRM-MLP. Decomposing the +4.85 budget-matched oracle into state discovery versus action selection: keeping the verifier's triggered states but selecting the best candidate there yields +0.96 SPL, while letting the oracle choose states and keeping the verifier's action choices yields +0.33 SPL. Relative to the oracle headroom, we define the ratios $(G_{\text{hybrid}} - G_{\text{v}})/(G_{\text{o}} - G_{\text{v}})$ (not $G_{\text{hybrid}}/G_{\text{o}}$): the action-selection ceiling is 18.2% and the state-discovery ceiling is 4.9%. Both capacities fail: the verifier captures 2.0% overall—the central open gap. The risk–coverage analysis (Figure 2) shows the verifier's incurred harm stays at or below 0.63 on the post-hoc confirmation sweep across coverages, while its net gain is near zero and non-monotonic in coverage as the budget-matched oracle grows monotonically—the score-gap ordering fails to rank the high-utility interventions it is allowed to take.

Pre-registered criteria. ForkPRM passed the scene-stability criterion and failed effect magnitude, oracle capture, cross-checkpoint transfer, and the improvement threshold over the pilot model (+0.32 observed vs. ≥ 0.5 required). We therefore interpret the calibration-split ordinary-ranking gains as genuine but narrow, and the confirmation as providing no evidence of practically substantial or policy-general improvement.

5.4 RQ4: How does verification scale with candidates and policy?

Figure 2 plots oracle and verifier gains as the candidate budget K grows from 2 to 4. Oracle headroom rises steeply (the best candidate is often ranked 3rd–4th by the policy: the two largest confirmation gains both came from rank-3 candidates), and verifier gains move in the same direction but by much smaller steps. Across fork positions, late decisions are both the most predictable and where the verifier concentrates its triggers; zero-shot application to the weaker checkpoint is the clearest failure mode, indicating the verifier still partially encodes its source policy's habits rather than consequence alone.

6 Discussion and Limitations

ForkPRM's labels are collected automatically from 20559 audited rollouts, improve ordinary pairwise ranking over unpaired outcome regression and over a cross-state permutation control, and train a verifier that remains positive under the scene-level sensitivity check. The same evaluation bounds the method: branch-selection gains remain far below the budget-matched oracle, and online gains and cross-policy generalization are not established. Three limits define the present scope: each state is supervised by a single rollout per arm, so within-state consequence variance is unmodeled; every split, confirmation included, draws on R2R training environments rather than val_unseen; and the deployed verifier has not been evaluated driving the policy over a full trajectory. The verifier–oracle gap is consistent with several causes—consequence representation, label noise, objective mismatch, and conservative calibration—that the present experiments do not separate.

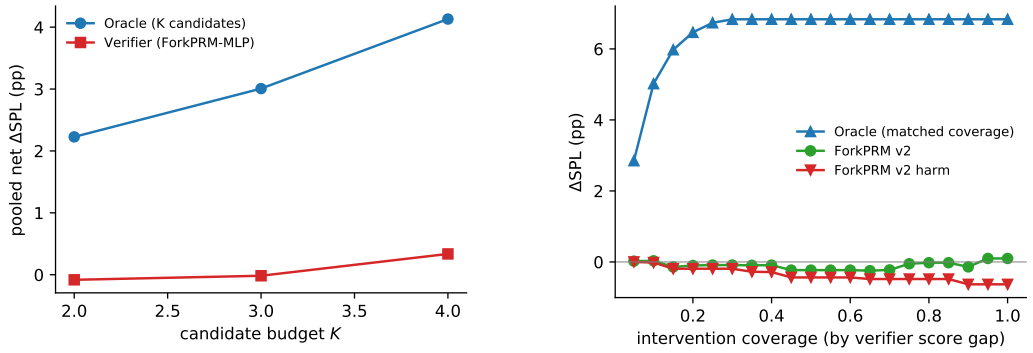


Figure 2: **Left: candidate-budget scaling** (pilot split, ForkPRM-MLP verifier—not the final ForkPRM). Oracle and verifier gains grow together with K; the verifier captures a small fraction. **Right: risk–coverage** on the confirmation split with the frozen ForkPRM verifier; a post-hoc descriptive sweep that was not used to select the frozen operating point. Gain and incurred harm as the intervention budget sweeps the score ranking; the oracle curve re-selects the same number of interventions by realized gain (budget-matched) at each coverage.

7 Conclusion

We presented ForkPRM, which turns environment branching into executable process supervision: candidate actions executed from a shared history under a fixed continuation policy yield baseline-relative process labels, which train a candidate-set verifier that requires no additional forks at application time. Shared-prefix supervision measurably improves ordinary consequence ranking, and the same evaluation locates the open problem between verifier ranking and policy utility. The technical lesson is that an embodied simulator is a usable process annotator, and that closing this gap is the open problem for embodied process verification.

Reproducibility Statement

All experiments use public assets (R2R, VLN-CE, MP3D, ETPNav checkpoints). Selection manifests, code hashes, model locks, per-episode records for every split (including the one-shot confirmation), and a rollout-count manifest (20559 audited rollouts in total: 13320 pilot, 555 feature-recapture replays, 2076 calibration, 4608 confirmation) are retained and will be released with the paper.

Generative AI use

Generative AI use. Generative AI tools assisted with literature search, research ideation and critique, experimental and statistical code, figure preparation, and manuscript editing. The authors reviewed the generated artifacts, reran and audited the experiments, verified the reported results and citations against primary sources, and take full responsibility for the submission.

References

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of llm outputs. *arXiv preprint arXiv:2407.13692*, 2024.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of classification-based policy iteration algorithms. *Journal of Machine Learning Research*, 17(19):1–30, 2016.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations (ICLR)*, 2019.

Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. Hop: History-and-order aware pre-training for vision-and-language navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

Table 4: **Data splits, counts, and audited rollouts (from the released rollout-count manifest). The 13320 pilot rollouts cover 555 routes $\times$ 3 forks $\times$ 4 arms $\times$ 2 checkpoints; the 1,128 fork states are the eligible (episode, fork) pairs among the $555 \times 3{=}1{,}665$ units, each supervised jointly by its four arms' rollouts.**

Split	Scenes	Routes	Fork states	Rollouts	Checkpoints
Pilot (train + held-out roles)	37	555	1,128 eligible	13,320	iter12000, iter4000
Feature recapture	29+8	555	same states	555	iter12000
v2 calibration	8	173	361 eligible	2,076	iter12000
Confirmation	12	192	409/384 eligible	4,608	iter12000, iter4000
Total	—	—	—	20559	—

Table 5: **Per-seed pairing control** (calibration; ordinary pairwise accuracy). The shared-prefix and cross-state-permutation variants use the identical pipeline; differences are positive for every seed (mean +7.6 pp, sd 1.4 pp, range [+5.6, +9.1] pp).

Seed	Shared-prefix	Permutation control	Difference
Seed 1	0.707	0.634	+0.0732
Seed 2	0.701	0.644	+0.0563
Seed 3	0.719	0.649	+0.0695
Seed 4	0.715	0.624	+0.0910
Seed 5	0.712	0.624	+0.0885
Mean	0.711	0.635	+7.6 pp

A Data and Counterfactual Rollout Protocol

Fork mechanics. Fork replay is deterministic per episode (fixed seed; deterministic kernels), and cross-arm prefixes are verified by hashing observations, poses, graph state, and RNG state at the fork. Each arm executes exactly one legal override at its fork and the frozen policy resumes afterwards.

Rollout accounting. The released count manifest lists 20559 audited rollouts across splits: 13320 pilot, 555 baseline-only feature-recapture replays (verifying bit-identical prefixes against the pilot states), 2076 calibration, and 4608 confirmation. Collection used up to three GPUs in parallel per phase.

B Model and Optimization Details

Architecture. Verifier trunk: two-layer MLP (width 128) over the candidate-specific input block (778 policy features plus 768 candidate-conditioned visual features per candidate); shared state enters through the invariant mean context over valid candidates. Heads consume $[h_i, c, h_i - h_0]$ (width 64). The full model has $\sim$0.9M parameters; inference is a single forward pass over at most four candidates with no environment interaction.

Training and ensemble. 300 epochs full-batch Adam (lr 1e-3, weight decay 1e-4); loss weights $\lambda_{\text{out}}{=}0.5$, $\lambda_{\text{rescue}}{=}\lambda_{\text{harm}}{=}0.3$, $\lambda_{\delta}{=}0.25$; harm-aware weight $w_i = 1 + 2.0\ \text{ReLU}(-y^{i,0})$; five seeds per fork, ensembled by averaging scores. Each seed trains in minutes on a single CPU core.

C Calibration and Statistical Evaluation

Threshold freezing. Gate search: margin quantiles $\{0, 0.3, 0.5, 0.7, 0.85, 0.925\} \times \tau \in \{-0.2, 0, 0.2\}$ per fork, selected on the calibration split and frozen before the confirmation run.

Branch-selection semantics. The confirmation evaluation is fully offline. For each of the 576 (episode, fork) units, all four single-override branches were executed once under deterministic replay; the frozen verifier scores the pre-fork state, and the realized outcome of a choice

Table 6: **Frozen inference thresholds** (selected on calibration only, before the confirmation run). The gate overrides only when the ensemble score margin over the policy action exceeds the margin and the predicted-success gap exceeds tau.

Fork index	Enabled	Frozen operating point
2 (early)	no	disabled
5 (middle)	yes	margin 3.61, tau 0.0
9 (late)	yes	margin 0.00, tau 0.0

Table 7: **Statistical procedures.** All confidence intervals are scene-level cluster bootstraps with multiplicity (10,000 resamples, fixed seed), matching the episode-weighted point estimand; a scene-balanced variant is reported in the main text where relevant.

Procedure	Setting
Cluster bootstrap	scenes, with multiplicity; 10,000 resamples
Primary estimand	episode-weighted ΔSPL (SPL points)
Secondary estimand	scene-balanced mean per scene
Sensitivity check	leave-one-scene-out (dependent, descriptive)
Seed ensembling	5 seeds, score-averaged; thresholds on calibration

is the corresponding branch's fully executed trajectory: `delta = SPL(arm=choice) − SPL(arm=baseline)`. Each unit contributes at most one override; the three forks of an episode are evaluated on independent replays of the *policy's own* history, so corrections do not compound and no verifier-in-the-loop trajectory is ever executed. The reported quantity is therefore an offline logged-prefix branch-selection value, not a closed-loop policy gain. Formally, with episodes $e{=}1,\dots,N$ ($N{=}192$), forks $f \in \{2,5,9\}$, and verifier choice $c(e,f) \in \{0,\dots,K{-}1\}$ (0 = keep the policy action):

$$\delta(e,f) = \begin{cases} U(\tau_{e,f}^{c(e,f)}) - U(\tau_{e,f}^{0}), & \text{unit eligible, fork gate enabled, } c(e,f) \neq 0, \\ 0, & \text{otherwise (ineligible, disabled fork, or no trigger),} \end{cases} \tag{3}$$

$$\hat{G} = \frac{1}{3} \sum_{f \in \{2,5,9\}} \frac{1}{N} \sum_{e=1}^{N} \delta(e,f). \tag{4}$$

Each $\tau_{e,f}^{a}$ is the fully executed single-override rollout of arm a at fork f of episode e; the denominator is always the full episode count N, so ineligible units and disabled forks contribute exact zeros. $\hat{G}$ is the per-fork, per-episode average of the single-override branch difference.

D Additional Results

Component ablations. The complete single-component ablation is Table 2 in the main text; the deployment comparison between the full package and the unstructured ForkPRM-MLP is Table 3.

Scaling and coverage. Candidate-budget scaling ($K \in \{2,3,4\}$) is Figure 2, left; the risk–coverage sweep with the frozen verifier is Figure 2, right.

Per-scene detail. Table 8 lists per-scene branch-selection detail.

Pilot and transfer. The pilot held-out evaluation, the full confirmation results, and the zero-shot second-checkpoint numbers are reported in Sections 5.1 and 5.3 and Table 3.

E Preregistered Decision Criteria

The five criteria, thresholds, and outcomes were frozen and hash-locked before the confirmation set was executed; the confirmation run was performed exactly once.

Table 8: **Per-scene branch-selection detail** on the confirmation split (frozen verifier; scene identifiers are MP3D scan ids). Net ΔSPL is the scene's contribution to the episode-weighted pooled estimand.

Scene	Routes	Trig.	Rescue	Harm	Net ΔSPL
1LXtFkjw3qL	16	4	0	0	+0.15
2n8kARJN3HM	16	4	1	0	-0.77
E9uDoFAP3SH	16	6	0	0	+0.40
PX4nDJXEHrG	16	1	0	0	+0.00
VVfe2KiqLaN	16	1	0	0	+0.37
VzqfbhrpDEA	16	3	0	0	+1.04
cV4RVeZvu5T	16	2	0	1	-1.43
jh4fc5c5qoQ	16	3	0	0	+0.95
kEZ7cmS4wCh	16	4	2	0	+4.93
qoiz87JEwZ2	16	4	0	1	-2.36
s8pcmisQ38h	16	3	0	1	-2.28
sKLMLpTHeUy	16	4	0	0	+0.19

Table 9: **Pre-registered submission criteria and one-shot outcomes.**

Criterion	Threshold	Outcome
Branch-selection gain	$\geq +1.0$ SPL, CI excl. 0	+0.10 (CI incl. 0): failed
Oracle capture	$\geq 25\%$	2.0%: failed
Scene stability (LOO)	$\geq 10/12$	12/12: passed
Second checkpoint	≥ 0	-0.05: failed
Gain over ForkPRM-MLP	$\geq +0.5$	+0.32: failed

F Reproducibility Checklist

- **Code and data.** Selection manifests, code hashes, model locks, per-episode records for every split (including the one-shot confirmation), and the rollout-count manifest are retained and will be released with the paper.
- **Compute.** Verifier training runs on CPU in minutes per seed; rollouts were collected on up to three commodity GPUs per phase.
- **Assets.** R2R, VLN-CE, Matterport3D, Habitat, and the ETPNav checkpoints are public; the frozen policy checkpoint hash is recorded in the released manifest.
- **Determinism.** Fork replay is seeded per episode and prefix-verified by hashing; the five-seed ensemble, calibration thresholds, and bootstrap seed are fixed in the released lock file.