AGENTIC DAGGER: SELF-IMPROVING INTERACTIVE IMITATION LEARNING VIA AGENTIC HARNESS

Anonymous authors
Paper under double-blind review

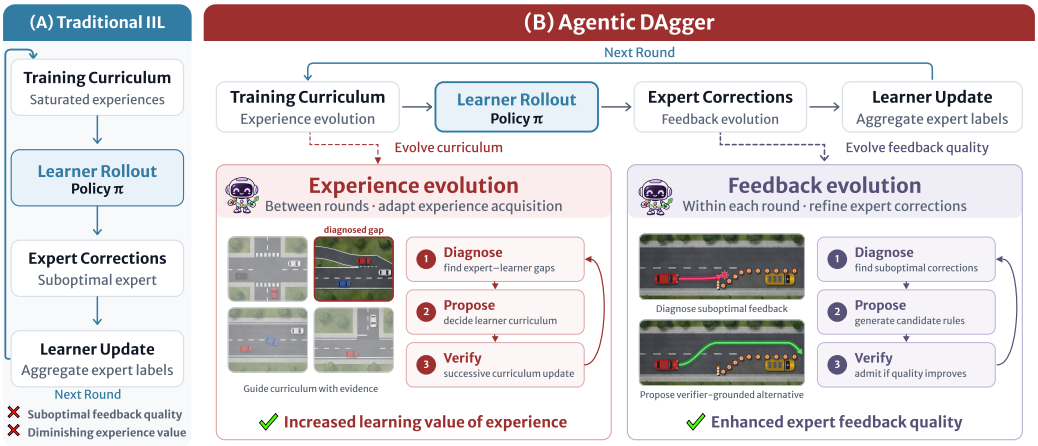


Figure 1: Conventional IIL and Agentic DAgger. **(A)** Traditional IIL updates the learner while suboptimal feedback and less informative experience impose learnability bottleneck. **(B)** Agentic DAgger retains the standard DAgger update but adds two harness-controlled loops. Within each round, feedback evolution improves verified feedback quality. Between rounds, experience evolution uses round evidence to improve the learning value of experience. These loops evolve the *closed-loop* training process alongside the learner update, and support sustained learner self-improvement.

ABSTRACT

Interactive imitation learning (IIL), such as DAgger, improves a policy by collecting expert feedback on learner-visited states. However, conventional IIL typically relies on a static feedback-generation pipeline comprising a likely suboptimal expert and training experience with diminishing learning value as the policy improves. Consequently, further policy improvement can be bottlenecked by both the quality of expert feedback and the utility of accumulated experience. We introduce **Agentic DAgger**, a training harness that jointly improves feedback quality and the learning value of experience alongside the learner. Agentic DAgger employs two specialized agent roles operating through *verifier*-grounded agentic loops. Within each training round, a *feedback-evolution loop* diagnoses original expert proposal, propose refined corrective labels, and verifies the labeling quality. Between rounds, an *experience-evolution loop* analyzes the learner-expert gap on the current training experience and acquires novel training experiences that maximize learning value. Together, these loops transform conventional IIL into a *self-improving training process that progressively improves corrective feedback and the learning value of training experience*. We evaluate Agentic DAgger by improving pretrained policies on NAVSAFE-∞, a verifiable closed-loop driving benchmark. Our experiments show that Agentic DAgger achieves higher closed-loop driving success than conventional IIL baselines under matched training budgets. It further improves learner performance across successive IIL rounds, demonstrating sustained policy self-improvement. These findings demonstrate the effectiveness of Agentic DAgger and highlight its potential to sustain policy self-improvement.

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.

1 INTRODUCTION

Interactive imitation learning (IIL) improves a policy through a *closed-loop* training process by collecting expert feedback on states the learner visits, thereby enabling supervision to track the learner's changing state distribution (Ross et al., 2011; Kelly et al., 2019; Zhang & Cho, 2017; Menda et al., 2019; Hoque et al., 2022; 2021; Cai et al., 2025a). However, conventional IIL cycle can remain bottlenecked by suboptimal expert feedback and the diminishing learning value of training experience. When feedback generation and experience acquisition remain fixed, these bottlenecks can limit sustained policy self-improvement despite continued interaction and learner updates.

In this paper, we investigate how to overcome these bottlenecks to sustain policy self-improvement. Our central idea is to introduce system-level adaptation around this closed-loop training process, jointly evolving feedback generation and experience acquisition. Inspired by recent advances in agentic harnesses (Huang et al., 2026; Xiao et al., 2026; Zhang et al., 2026a; Chen et al., 2026), we introduce **Agentic DAgger**, a training harness that draws on evidence accumulated throughout the IIL cycle and coordinates tool use, memory, and verification to improve corrective feedback and the learning value of training experience alongside the learner, as illustrated in Figure 1.

The harness implements two agentic loops for refining expert feedback and adapting training experience, each following a *diagnose–propose–verify* procedure. Within each round, the *feedback-evolution loop* diagnoses suboptimal expert feedback, proposes refined corrections, and verifies their quality before accepting the change. Between rounds, the *experience-evolution loop* diagnoses expert–learner gaps, proposes curriculum changes, and verifies their benefit through learner training and evaluation. For the subsequent rounds, the accepted feedback refinements and curriculum changes are provided for learner to perform the standard DAgger-style update.

We study Agentic DAgger on a verifiable closed-loop driving benchmark (Bao et al., 2026), and our experiments yield three main findings. First, repairing suboptimal expert corrections yields more useful supervision, improving learner recovery and reactivity beyond conventional IIL under matched training budgets (Section 4.2). Second, coordinating feedback and experience evolution sustains policy improvement across rounds by tracking changes in expert–learner gaps and adjusting scenarios where learners have the most room for improvement (Section 4.3). Third, reliable self-improvement requires mitigating objective hacking, overcoming search stagnation, and addressing potential objective mismatch (Section 4.4). Together, these findings demonstrate the effectiveness of Agentic DAgger and highlight its potential to enable sustained policy self-improvement.

Our contributions are three-fold:

1. We formulate an agentic extension of IIL that adapts corrective feedback and experience acquisition alongside the learner, and introduce **Agentic DAgger**, a training harness that enables policy self-improvement by enhancing expert feedback and the learning value of training experience.
2. We develop two agentic loop procedures within the IIL cycle: a *feedback-evolution loop* that diagnoses gaps in expert feedback quality and refines corrections through iterative proposal and verification within training rounds, and an *experience-evolution loop* that diagnoses expert–learner gaps and proposes curriculum changes between IIL training cycles.
3. We evaluate Agentic DAgger on NAVSAFE-∞ (Bao et al., 2026), a verifiable closed-loop driving benchmark, where it outperforms the strongest conventional IIL baseline under matched training budgets and sustains policy self-improvement across IIL learning cycles.

2 RELATED WORK

Interactive imitation learning. IIL supports sample-efficient policy learning by providing direct expert feedback to learner-visited states (Kelly et al., 2019; Hoque et al., 2022; Cai et al., 2025a). DAgger mitigates covariate shift by aggregating these corrections across training rounds (Ross et al., 2011). Human- and robot-gated variants balance supervision cost and safe exploration by adapting when to intervene or query (Kelly et al., 2019; Menda et al., 2019; Hoque et al., 2022; 2021; Cai et al., 2025a;b). Even with adaptive querying or expert selection, useful feedback depends on the corrections available supervisors can produce: suboptimal experts may fail in unfamiliar environments,

leaving gaps that additional queries alone cannot resolve (Cai et al., 2025a;b). Agentic DAgger addresses this bottleneck via feedback-evolution loop through iterative proposal and verification of refined corrective actions to improve expert label quality.

Autoresearch and automated curriculum design. Autoresearch motivates our experience-evolution loop by framing training-data design as an iterative process of proposing interventions and testing their effect on learner performance. AutoData generates synthetic data and assesses its value through model training (Kulikov et al., 2026), while RSIBench-Data fixes training and evaluation services to study iterative data research, showing that later revisions can undo earlier gains (Meng et al., 2026). Automated curriculum methods generate, replay, or evolve tasks according to regret or estimated learning potential (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022), while DataMIL estimates the downstream value of robot imitation data (Dass et al., 2026). With those motivations, Agentic DAgger's experience-evolution loop analyzes each round's learner's failures and evolves a useful curriculum for the learner's next round update.

Self-Improving agents. Self-improving agent systems turn execution experience into persistent updates to model parameters, prompts, memory, tools, and control logic (Ren et al., 2026). ENPIRE uses robot rollouts to guide revisions to training infrastructure and algorithm code (Xiao et al., 2026), while ASPIRE consolidates validated repairs into reusable skills (Lu et al., 2026). The Darwin Gödel Machine evolves its own codebase through an archive of agents (Zhang et al., 2026b), and AVO uses coding agents to propose, repair, and verify variations within evolutionary search (Chen et al., 2026). Agentic DAgger applies this principle of persistent, evidence-driven adaptation to the IIL supervision process, forming a self-improvement loop in which the policy and its supervision process evolve together through the harness design.

3 Agentic DAgger

3.1 Preliminaries: Problem Setting and the IIL Cycle

Problem setting. Starting from an offline-pretrained learner π_0^L, our objective is to improve closed-loop performance via an expert that provides feedback in the form of corrective actions at learner-visited states. These corrections supply supervision for fine-tuning the learner (Ross et al., 2011; Kelly et al., 2019; Hoque et al., 2022).

Interactive imitation learning cycle. Standard IIL alternates between collecting learner rollouts with expert feedback and training the learner on the accumulated supervision. At round r, learner π_r^L collects rollouts under curriculum $c_r = (p_r, \mu_r)$: p_r is the scenario-sampling distribution, and μ_r specifies how training examples are sampled, weighted, or replayed. Two operators determine expert feedback within rounds and the curriculum between rounds:

$$y_{r,t}^E = \mathrm{Feedback}(\mathcal{C}_{r,t}^{\mathrm{fb}}); \qquad c_{r+1} = \mathrm{Curriculum}(\mathcal{C}_{r+1}^{\mathrm{exp}}), \tag{1}$$

where the *feedback operator* $\mathrm{Feedback}$ maps the interaction context $\mathcal{C}_{r,t}^{\mathrm{fb}}$, including the current state and learner behavior, to an expert corrective action $y_{r,t}^E$ when feedback is requested at step t. By default, this action replaces the learner's proposed action during rollout. It serves as the training label for that state only after meeting the label-quality requirements. The *curriculum operator* $\mathrm{Curriculum}$ maps the post-update training and evaluation context $\mathcal{C}_{r+1}^{\mathrm{exp}}$ to c_{r+1}, determining which scenarios to sample and how to use supervision in the next round.

3.2 Harness Overview

Agentic DAgger is a training harness around the IIL cycle described in Section 3.1. Figure 1 provides an overview and compares it with conventional IIL. It addresses suboptimal feedback and less informative experience by replacing the two operators in Equation 1 with tool-using agentic loops while preserving the standard DAgger learner update:

$$y_{r,t}^E = \mathrm{Agent}_{\mathrm{fb}}(\mathcal{C}_{r,t}^{\mathrm{fb}}); \qquad c_{r+1} = \mathrm{Agent}_{\mathrm{exp}}(\mathcal{C}_{r+1}^{\mathrm{exp}}), \tag{2}$$

where each Agent denotes the complete loop of proposal, evaluation, and selection under the shared *orchestrator*. The *feedback-evolution agent*, $\mathrm{Agent}_{\mathrm{fb}}$, runs within rounds: it diagnoses suboptimal

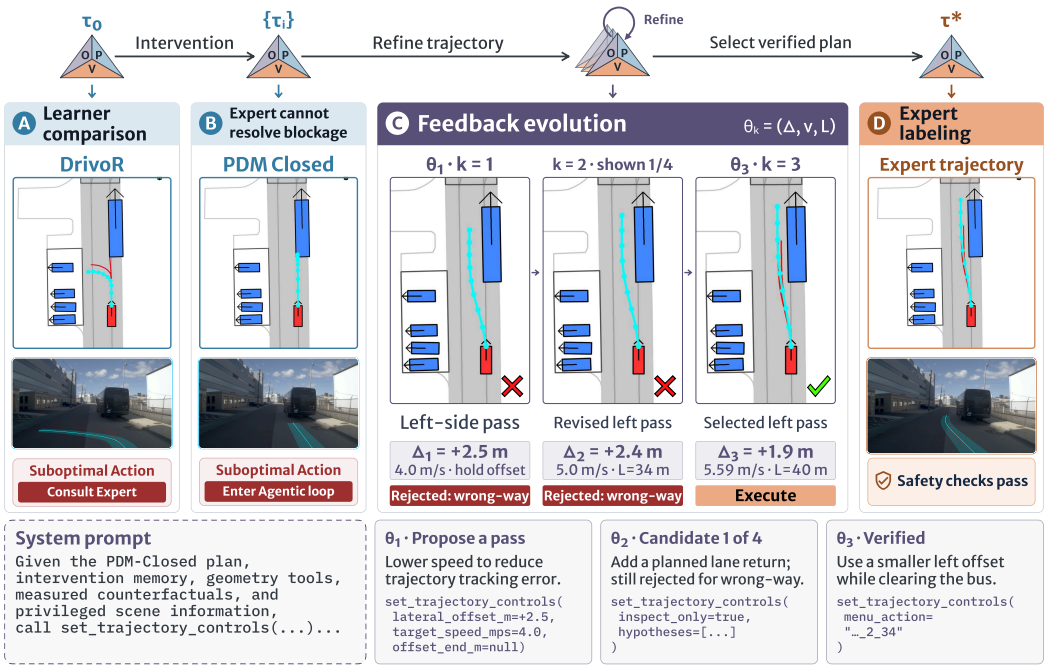


Figure 2: **Illustration of the feedback-evolution loop.** *(a) Learner Failure:* The learner predicts a driving trajectory from scene observations which triggers intervention with the expert, PDM-Closed. *(b) Expert Intervention:* The suboptimal expert cannot generate a trajectory that makes sufficient progress past the stopped bus so it enters the inner feedback evolution loop. *(c) Verifier-Grounded Feedback-Evolution Loop:* Using privileged scene information and *verifier* feedback, the agent iteratively adjusts lateral offset, speed, and lane re-entry to find a verified trajectory. *(d) Expert Labeling:* The final verified trajectory serves as an expert label for supervising the learner.

expert corrections, proposes refined corrective actions, and revises them using rule-based verifiable feedback. The *experience-evolution agent*, $\text{Agent}_{\text{exp}}$, runs between rounds: it diagnoses gaps in learner performance and proposes changes to curriculum for next learner update. Sections 3.3 and 3.4 elaborate these two agentic loops, and Appendix A provides the full harness details.

3.3 Agentic Feedback-Evolution Loop

Feedback quality. Feedback evolution seeks to improve corrective supervision through iterative refinement of expert actions at learner-visited states. A suboptimal expert may provide valid corrections while leaving more effective recovery actions unexplored. For example, PDM-Closed (Dauner et al., 2023) constructs trajectories using a predefined grid of lateral offsets and target-speed policies. In Figure 2, its proposal makes insufficient progress past a stopped bus, motivating the agent to inspect alternative bypasses and refine their speed and lane re-entry.

Loop procedure and evolution criteria. The agentic loop coordinates an *Orchestrator*, *Proposer*, and *Verifier* to improve expert corrections through iterative proposal and verification. It is triggered when the base expert action fails the *Verifier*'s acceptance check. The *Orchestrator* then assembles the feedback context $\mathcal{C}^{\text{fb}}_{r,t}$ for the *Proposer*. The *Proposer* and *Verifier* use agent tools to propose and evaluate candidate corrections under the same scene conditions. The *Orchestrator* iteratively coordinates the *Proposer* and *Verifier* to refine candidates using *verifier* feedback until a correction passes verification, and then returns it as $y^{E}_{r,t}$.

Agent tools. The *Verifier* uses trajectory-scoring tools to assess action quality by aggregating selected subscores of the Extended Predictive Driver Model Score (EPDMS) (Cao et al., 2025): no at-fault collisions (NC), drivable-area compliance (DAC), driving-direction compliance (DDC), traffic-light compliance (TLC), ego progress (EP), time to collision within bound (TTC), lane keeping (LK),

and history comfort (HC). We exclude extended comfort (EC) from the internal *verifier* objective to mitigate objective hacking (Section 4.4). The *Proposer* uses tools that expose privileged simulator states and front-camera observations, together with counterfactual trajectory refinement tools for adjusting lateral offsets, target speeds, and lane re-entry to prevent search stagnation according to the feedback context.

3.4 Agentic Experience-Evolution Loop

Learning value of experience. Experience evolution seeks to improve the learning value of the curriculum as the learner improves and expert corrections become more effective. Repeated training can reduce the learner's remaining errors on familiar scenarios, leaving fewer opportunities for useful supervision. Meanwhile, feedback evolution can make previously unproductive scenarios informative by enabling better corrections. These observations motivate replacing mastered scenarios with those in which the learner still struggles and the expert can provide effective corrections.

Loop procedure and evolution criteria. The agentic loop coordinates an *Orchestrator*, *Proposer*, and *Verifier* and operates between IIL rounds, using evaluations of the accepted learner to guide curriculum refinement. The *Orchestrator* assembles capability reports from evaluation benchmarks, learner-expert reports from scenario pool, and previous training round outcomes into the experience context $\mathcal{C}_{r+1}^{\text{exp}}$ for the *Proposer*. The *Proposer* and *Verifier* use agent tools to propose curriculum changes and evaluate their effects on learner performance. Acceptance requires a valid trial and a strict improvement in aggregate success rate on the development cohort. The *Orchestrator* uses trial outcomes to guide subsequent proposals. When the acceptance criteria are met, it promotes the candidate learner and adopts the proposed curriculum as c_{r+1}; otherwise, it retains the previously accepted learner and curriculum.

Agent tools. The *Orchestrator* uses collection and training tools to gather expert feedback under the proposed curriculum and train a candidate learner from the accepted parent checkpoint. The *Verifier* uses evaluation tools to compare candidate and parent success rates on the same development cohort. The *Proposer* uses tools to inspect capability reports, such as benchmark results and learner's ability scores, learner-expert gap report on scenario pool (see Figure 6), modify curriculum. The learner-expert gap report is conducted via a *rollout probe* where expert and learner would be rolled out in scenario pool to determine their performance gap.

4 Experiments

In this section, we evaluate the central hypothesis that evolving feedback and experience can expand the useful supervision available to an improving learner throughout the IIL cycle. Specifically, we answer the following research questions (RQ). **RQ1:** Does feedback evolution provide better corrective supervision than existing IIL baselines and improve the learner's closed-loop performance? **RQ2:** Do feedback and experience evolution together enable progressive self-improvement of the learner across rounds? **RQ3:** Which design choices support reliable feedback and experience evolution throughout the IIL cycle?

4.1 Experimental Setup

Benchmarks and evaluation settings. We evaluate policy improvement on NavSafe-∞ (Bao et al., 2026), a verifiable closed-loop driving benchmark containing 280 safety-critical scenarios. The goal is to improve the closed-loop driving performance of a pretrained policy from NAVSIM benchmarks (Dauner et al., 2024). Following NavSafe-∞, the primary metrics are success rate (SR), Driving Score (DS), Driving Efficiency (DE) and Comfort, and four category-level capability scores: Traffic Crashes (TC), Vulnerable Road User Crashes (VRUC), Traffic Violations (TV), and Traffic Incidents (TI). We refer readers to NavSafe-∞ for more evaluation details.

Agentic DAgger configuration. We use DrivoR (Kirby et al., 2026) as the learner, with a reference collection budget of 100 episodes per cycle. The collection data is from Nuplan (Caesar et al., 2021) validation split that is disjoint from NavSafe-∞ scenarios. Agentic DAgger (Expert) denotes

Table 1: Expert performance and learner improvement on NAVSAFE-∞. Each learner + row applies the indicated IIL method independently to pretrained DrivoR. Agentic DAgger enables feedback evolution with experience evolution disabled. $\star$ denotes privileged simulator access and † denotes human control.

Method	Closed-loop metrics				Closed-loop ability scores			
	DS	SR	DE	Comf.	TC	VRUC	TV	TI
Human reference								
Human driver†	100.00	100.00	316.52	75.15	100.00	100.00	100.00	100.00
Privileged models								
PDM-Closed*	90.17	81.29	305.27	82.49	88.97	71.62	84.24	57.78
Agentic DAgger (Expert)*	100.00	100.00	295.79	83.99	100.00	100.00	100.00	100.00
Pretrained E2E Policy Baseline								
DrivoR	64.74	50.69	188.02	87.92	58.61	31.38	52.53	43.33
Standard IIL								
+ DAgger	67.61	65.00	200.13	95.71	68.67	31.67	79.70	43.33
Robot-gated IIL								
+ EnsembleDAgger	67.21	57.86	185.53	97.70	68.67	31.67	61.52	43.33
+ ThriftyDAgger	69.87	61.43	174.36	99.52	78.67	56.67	61.52	10.00
+ AIM	67.21	57.86	183.25	89.87	68.67	31.67	61.52	43.33
Human-gated IIL								
+ HG-DAgger	68.58	52.02	169.38	98.47	44.00	26.67	70.91	43.33
+ PVP	68.91	45.48	126.72	73.54	51.33	27.50	48.79	37.78
+ PPL	69.33	67.14	200.64	80.73	73.33	51.67	75.15	37.78
+ Agentic DAgger	**80.07**	**69.05**	**201.03**	66.72	**86.67**	50.00	66.67	**44.44**

the expert agent traversing scenarios with privileged simulator information. The agent backend is Claude Opus 5 (Anthropic, 2026). We refer reader to Appendix B for more details.

4.2 RQ1: Feedback Quality and Closed-Loop Policy Improvement

Experiment protocol. In this section, in order to understand the contribution of feedback evolution to corrective supervision and downstream learner performance, we disable experience evolution loop for curriculum changes. Under matched training budgets, we compare Agentic DAgger with standard DAgger (Ross et al., 2011); robot-gated methods EnsembleDAgger (Menda et al., 2019), ThriftyDAgger (Hoque et al., 2022), and AIM (Cai et al., 2025a); and human-gated methods HG-DAgger (Kelly et al., 2019), PVP (Peng et al., 2023), and PPL (Cai et al., 2025b).

Feedback evolution improves expert corrections. Agentic DAgger (Expert) achieves a higher closed-loop success rate than PDM-Closed in Table 1. Their closed-loop success rates measure episode-level expert competence and provide supporting evidence for the feedback-evolution loop. Figure 3(a) complements this comparison with corrective action sequences from the same learner-induced state, illustrating how different experts respond to a failure encountered during on-policy collection episode. By refining the base expert's proposals, feedback evolution can produce effective recoveries in cases where PDM-Closed or human corrections fall short, providing more useful supervision at safety-critical states visited by the learner.

Improved corrections strengthen learner recovery and reactivity. Table 1 demonstrates learner's performance fine-tuned by Agentic DAgger. Figure 3(b) compares the pretrained and updated learners under same initial conditions. We observe that through the on-policy correction, the finetuned baseline exhibits substantial improvements from two aspects of closed-loop behavior, namely recovering from policy-induced errors (*recovery*) and responding to changing hazards (*reactivity*).

Comparison with IIL baselines. Table 1 compares learner improvements across standard, robot-gated, and human-gated IIL methods. Agentic DAgger achieves greater improvements in closed-

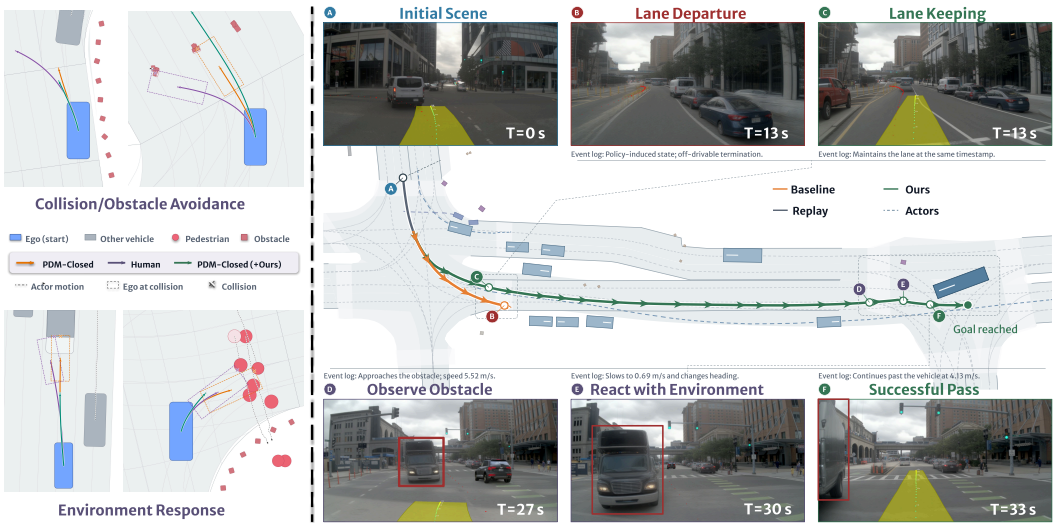


Figure 3: **Feedback quality and closed-loop policy improvement.** *Left:* Comparisons of different expert labeling qualities. Executed trajectories under Human, PDM-Closed and Agentic-DAgger (Expert) corrections from the same learner-induced state. Our feedback-evolution loop provides safer action labels compared with other experts. *Right:* Illustration of closed-loop policy improvements. With our method, the pretrained policy shows robustness of safety behaviors under policy-induced states and react with environment changes.

loop driving performance than the robot-gated baselines, particularly in the TC and TI categories. These results demonstrate the effectiveness of evolving expert corrections to improve supervision for safety-critical driving behaviors. Agentic DAgger also achieves performance comparable to human-gated IIL methods without requiring live human supervision during on-policy collection. Whereas human-gated methods rely on experts to monitor rollouts, intervene, and demonstrate corrective actions (Kelly et al., 2019; Peng et al., 2023), Agentic DAgger automates this process, avoiding the associated human time and intervention burden. These mechanisms enable continued corrective data collection under a fixed verification protocol without being constrained by human-expert availability.

4.3 RQ2: Progressive Learner Self-Improvement

Experiment protocol. We study progressive closed-loop policy improvement on the NavSafe-∞ benchmark, comparing standard DAgger, feedback-evolution only, experience evolution only, and full Agentic DAgger. All configurations start from the same pretrained policy and initial curriculum, with matched collection and learner-update budgets per executed training attempt. Standard DAgger and feedback-only training retain the initial training scenarios and continue from the latest checkpoint. Experience-only and full Agentic DAgger use NavSafe-∞ as development feedback for curriculum selection from a pool of 1,000 scenarios drawn from the nuPlan validation split, without agent access to the underlying benchmark scenes.

Single-loop adaptation extends improvement but eventually saturates. Figure 4 shows that standard DAgger, feedback-only training, and experience-only training plateau after cycles 2, 3, and 5, respectively. Both adaptation strategies therefore extend policy improvement beyond standard DAgger, but their gains eventually saturate after few cycles. Experience-evolution only training continues to propose curriculum changes through unsuccessful trials. This suggests that exposing the policy to different driving situations produces marginal gains when the available expert corrections no longer support further learning. Conversely, the feedback-evolution only plateau indicates that refining corrections within a fixed scenario set does not sustain improvement throughout training.

Joint evolution sustains policy self-improvement across training cycles. Full Agentic DAgger continues to improve beyond the saturation points of both single-loop variants, reaching a success rate of 76.78% by cycle 10, although gains become smaller in later cycles. Together, these com-

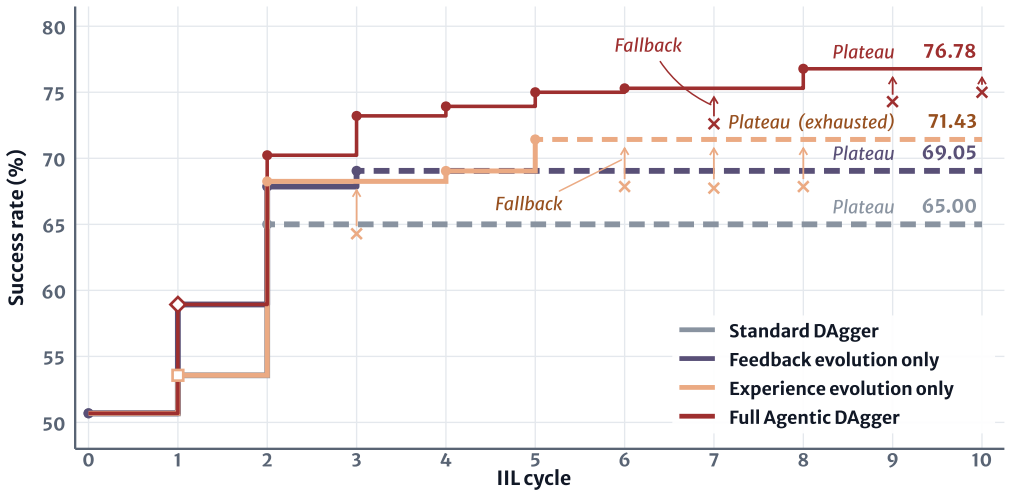


Figure 4: **Evolution trajectories analysis.** Learner success rate on NAVSAFE-∞ over 10 IIL cycles for standard DAgger, feedback-only training, experience-only training, and full Agentic DAgger.

parisons support treating scenario selection and corrective supervision as complementary parts of progressive policy self-improvement.

4.4 RQ3: Design Choices for a Reliable Self-Improving IIL Cycle

In this section, we examine design choices for reliable self-improvement, focusing on mitigating objective hacking and search stagnation in feedback evolution and guiding experience evolution with expert–learner gap evidence.

Mitigating objective hacking through *verifier* rule-based metric design. Since the *verifier* utilizes the aggregate scores of different rule-based metrics that examine different behavioral aspects, we find that the agent is prone to find a gap where it provides feedback that is not intended for recovery behavior. Figure 5(*Left*) illustrates how an agent can exploit a score component to pass the *Verifier*'s acceptance check while leaving the intended recovery behavior unresolved. In the driving environment, gains in extended comfort (EC) can compensate for reduced progress, allowing creeping corrections to receive high aggregate scores. Such a mismatch between the optimized score and the intended recovery behavior is characteristic of reward hacking (Skalse et al., 2022). We therefore exclude EC from the internal EPDMS selection objective, retaining the separate history-comfort term and the full EPDMS definition for evaluation. This prevents the agent produces undesirable recovery actions by exploiting comfounding metrics.

Counterfactual guidance for proposal search. Figure 5(*Right*) illustrates how proposal search can revisit similar trajectories or settle on a feasible correction while leaving better recovery alternatives unexplored. *Verifier* feedback identifies shortcomings of an evaluated candidate, but does not directly reveal which alternative maneuvers are feasible at the same state. We therefore supply counterfactual trajectory outcomes before the agent's first proposal. The harness instantiates alternative lateral and longitudinal maneuvers through the planner, scores them with the fixed *verifier*, and exposes their progress and safety outcomes to the agent. These comparisons provide concrete evidence for choosing recovery parameters. The generated proposed correction still passes through the usual verification and admission checks. Counterfactual guidance makes alternative recoveries and their consequences explicit before search, helping the agent move beyond repeated local revisions.

Addressing objective mismatch in experience acquisition. Capability labels and category-level performance provide an interpretable starting point for retrieving training scenarios, consistent with prior work on skill-based data generation and adaptive training (Khan et al., 2025; He et al., 2026). These labels guide where to search, but treating them as sufficient evidence of learning value risks an *objective mismatch* between experience selection and learner improvement. Figure 6 illustrates

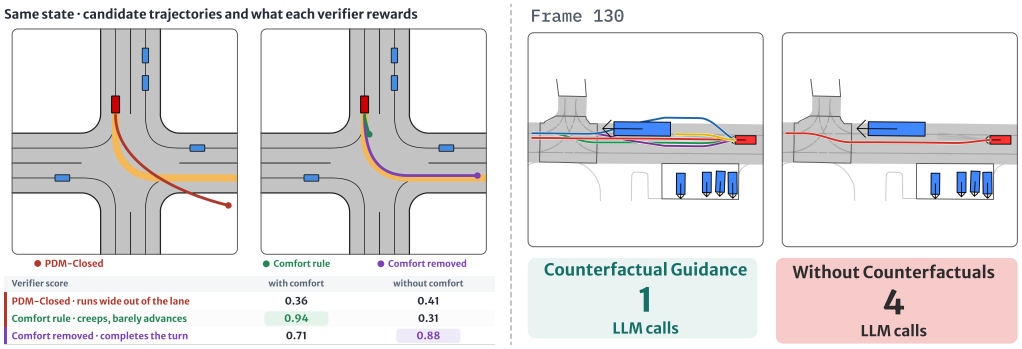


Figure 5: **Illustration of objective hacking and counterfactual-guided search.** *Left:* The LLM examines the EPDMS score breakdown and proposes a correction that exploits comfort, changing its outcome from fail to verified. The resulting trajectories achieve higher EPDMS while favoring conservative motion with limited ego progress. *Right:* From the same scene state, counterfactual trajectory alternatives guide proposal refinement, allowing the agent to identify a correction in one LLM call, compared with four calls without guidance.

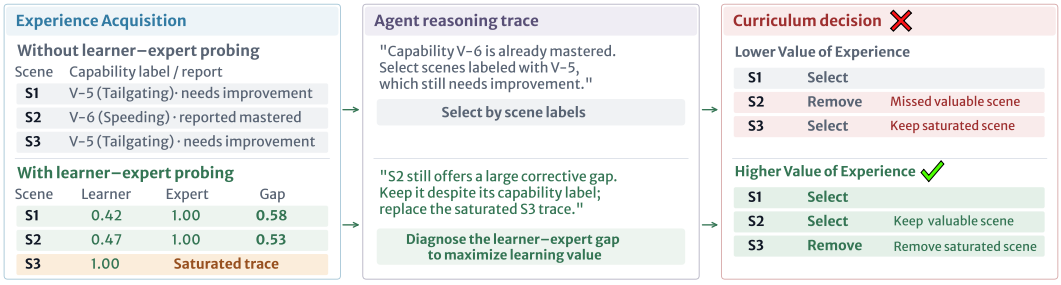


Figure 6: **Illustration of objective mismatch.** *Top:* Capability labels and reports provide an initial selection signal but potentially deviates true objective of minimizing expert–learner gap. *Bottom:* Learner–expert probing reveals a substantial corrective gap in S2 and a saturated S3 trace, motivating retention of S2 and replacement of S3.

this risk: label-based selection discards S2 because its capability is reported as mastered, despite a substantial expert–learner gap, while retaining S3 because its capability is reported as weak, although the learner already handles that scenario successfully. The harness therefore complements label-guided retrieval with rollout probes that assess scenario-level expert–learner gaps and the availability of useful corrective supervision in order to prevent potential objective mismatch.

5 Conclusion

We introduced Agentic DAgger, a training harness that improves expert feedback and the learning value of training experience alongside the learner. Through two coupled diagnose–propose–verify loops, the harness refines expert corrections within rounds and adapts training experience between rounds while retaining the standard DAgger learner update. Experiments in closed-loop driving demonstrate that these coordinated adaptations provide more effective supervision beyond conventional IIL baselines and sustain policy self-improvement across IIL cycles.

Limitations and Future Work. Our framework currently uses verifiable rewards in the training environment to guide feedback and experience evolution. An important direction for future work is developing alternative feedback and evaluation mechanisms for settings where such rewards are unavailable or do not fully capture the desired behavior.

AI USE STATEMENT

In this work, we have not used generative AI tools for tasks with required disclosure, such as "Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results". We used generative AI tools for assisted editing software code, searching for information, editing and restructuring of manuscripts, formatting references, and writing rendering code for manuscript figures. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study evaluates driving-policy improvement in simulation. Improved simulator scores do not establish real-world safety; deployment would require independent validation of the policy, verifier, and operating conditions. The human-involved experiments for human-gated IIL baselines are conducted following Human Subject Research Protocol described in Appendix C.

REPRODUCIBILITY STATEMENT

We plan to provide a GitHub repository in the future, including checkpoints, execution logs, implementation code, dependency versions, and README instructions for reproducibility. Appendix A demonstrates example system prompts, tools for the harness and Appendix B details the training and evaluation setup.

REFERENCES

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, 2026. Accessed 2026-09-25.

Yuxin Bao, Hongwei Ruan, Luobin Wang, Seth Z. Zhao, Ziyang Leng, Zihan Zhang, Yu Zeng, Rowan McAllister, Henrik Christensen, and Bolei Zhou. NavSafe-∞: Benchmarking closed-loop driving safety in photorealistic environments. arXiv preprint arXiv:2609.26618, 2026.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. In *CVPR Workshop on Autonomous Driving: Perception, Prediction and Planning (ADP3)*, 2021.

Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Robot-gated interactive imitation learning with adaptive intervention mechanism. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 6243–6256. PMLR, 2025a. URL https://proceedings.mlr.press/v267/cai25e.html.

Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Predictive preference learning from human interventions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4709–4722. PMLR, 2025. URL https://proceedings.mlr.press/v305/cao25a.html.

Terry Chen, Zhifan Ye, Bing Xu, Zihao Ye, Timmy Liu, Ali Hassani, Tianqi Chen, Andrew Kerr, Haicheng Wu, Yang Xu, Yu-Jung Chen, Hanfeng Chen, Aditya Kane, Ronny Krashinsky, Ming-Yu Liu, Vinod Grover, Luis Ceze, Roger Bringmann, John Tran, Wei Liu, Fung Xie, Michael

Lightstone, and Humphrey Shi. AVO: Agentic variation operators for autonomous evolutionary search. arXiv preprint arXiv:2603.24517, 2026.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. DataMIL: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations (ICLR)*, 2026.

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1268–1281. PMLR, 2023. URL https://proceedings.mlr.press/v229/dauner23a.html.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 37, pp. 28706–28719, 2024.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 13049–13061, 2020.

Yinghui He, Abhishek Panigrahi, Yong Lin, and Sanjeev Arora. STAT: Skill-targeted adaptive training. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/17d60fef592086d1a5cb136f1946df59-Abstract-Conference.html.

Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S. Brown, Daniel Seita, Brijen Thananjeyan, Ellen Novoseller, and Ken Goldberg. LazyDAgger: Reducing context switching in interactive imitation learning. In *IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 502–509, 2021.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 598–608. PMLR, 2022. URL https://proceedings.mlr.press/v164/hoque22a.html.

Yue Huang, Wenjie Wang, Han Bao, Yuchen Ma, Xiaonan Luo, Yi Nian, Haomin Zhuang, Zheyuan Liu, Yue Zhao, and Xiangliang Zhang. MemoHarness: Agent harnesses that learn from experience. arXiv preprint arXiv:2607.14159, 2026.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021. URL https://proceedings.mlr.press/v139/jiang21b.html.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083, 2019. doi: 10.1109/ICRA.2019.8793698.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. DataEnvGym: Data generation agents in teacher environments with student feedback. In *International Conference on Learning Representations (ICLR)*, 2025.

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on registers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Kirby_Driving_on_Registers_CVPR_2026_paper.html.

Ilia Kulikov, Chenxi Whitehouse, Tianhao Wu, Yixin Nie, Swarnadeep Saha, Eryk Helenowski, Weizhe Yuan, Olga Golovneva, Jack Lanchantin, Yoram Bachrach, Jakob Foerster, Xian Li, Han Fang, Sainbayar Sukhbaatar, and Jason Weston. Autodata: An agentic data scientist to create high quality synthetic data. arXiv preprint arXiv:2606.25996, 2026.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic /skills discovery for robotics. arXiv preprint arXiv:2607.00272, 2026.

Kunal Menda, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. EnsembleDAgger: A Bayesian approach to safe imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5041–5048, 2019. doi: 10.1109/IROS40897.2019.8968287.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. RSIBench-Data: Benchmarking data-centric research for recursive self-improvement. arXiv preprint arXiv:2607.25886, 2026.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Zhenghao Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 77969–77992, 2023. doi: 10.52202/075280-3406. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f57ffe47d0b528fbb97901d16bd4eba2-Abstract-Conference.html.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Nanbo Li, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. arXiv preprint arXiv:2607.13104, 2026.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 9460–9471, 2022.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic robot policy self-improvement in the real world. In *Conference on Robot Learning (CoRL)*, 2026.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves. arXiv preprint arXiv:2606.09498, 2026a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/aa5f5e6eb6f613ec412f1d948dfa21a5-Abstract-Conference.html.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. doi: 10.1609/aaai.v31i1.10857. URL https://ojs.aaai.org/index.php/AAAI/article/view/10857.

APPENDIX

The appendix is organized as follows.

- Appendix A: Harness details on Agentic DAgger, including system prompts, tools, agentic loops definitions and examples.
- Appendix B: Experiment details including training and evaluation setup for each method (Section 4).
- Appendix C: Human Subject Research Protocol.

A HARNESS AND IMPLEMENTATION DETAILS

In this section, we detail the core components of Agentic DAgger: the system prompts, the tools, and the agentic loop definition. The original harness traces and loop logs contain personally identifiable information, so we apply an anonymization step that removes these details while preserving the essential content and step structure.

A.1 SYSTEM PROMPT

Feedback-evolution agent prompt. The feedback agent refines the expert's proposed behavior when verification identifies safety, feasibility, or progress concerns. The resulting expert behavior supplies corrective supervision for the learner. Listing 1 shows the system prompt and Listing 2 shows the context template that feedback agent receives within the feedback-evolution loop.

Listing 1: Feedback-evolution agent system prompt

```
You are the trajectory-repair agent inside a closed-loop driving
harness. You make the final maneuver decision. The verifier
accepts or refuses the exact choice you submit, and the harness
never executes an unchosen higher-scoring alternative.

GOAL
- Complete the route safely. Prefer feasible progress toward the
  goal over a higher stationary score or perfect comfort.
- Wait or brake when a concrete current conflict, signal, or
  unresolved safety concern requires it. Do not assume that an
  actor will yield.

RULES OF RECORD (reported by the harness, never softened)
- Completion: remaining route <= max(2 m, 1% of the route) and
  the ego within 10 m of the route end. Waiting never completes
  a route.
- Success: completion with no contact, red-light crossing,
  drivable-surface departure, or wrong-way termination.
- Deadlock: speed below 0.1 m/s for 5 s ends the episode; a red
  light or stop-sign zone suspends this, but 120 s stationary
  ends the episode regardless.
- Any contact ends the episode; one footprint corner off the
  drivable surface for 0.5 s ends it.

EVIDENCE
- Read every supplied scene image. Treat verifier evidence as
  authoritative, but a score alone does not certify a maneuver:
  a stationary trajectory can score 1.0.
- Compare measured safe progress, clearance, pace, and
  continuation. A refused menu does not prove that every
  maneuver is infeasible.
- Before slowing or stopping, consider rear traffic as well as
  the actors ahead.
- Memory of recent interventions is advisory. Option IDs from
  earlier states cannot authorize a new plan.
- [Guidance on turns, passing blockers, lane-keeping repairs,
  and offset-window geometry omitted.]
```

```
ACTIONS (submit one)
- menu_action: choose a measured option; it is rechecked before
  execution.
- numeric controls: commit exact edits to the reference path
  (lateral offsets, window, anchor, target speed or acceleration).
- inspect_only: simulate up to <H> hypotheses, or a clearance
  request for a named actor, without executing anything.
- brake_to_stop: brake to rest along a chosen steering reference;
  checked for front and rear contact and road constraints.
- keep_planner_and_replan: keep the planner's trajectory, allowed
  only after a refused motion proposal or a complete menu with no
  admissible motion at this state.

Use refusal feedback to revise the plan rather than repeating an
unchanged hypothesis. Return one structured action, never code or
prose in its place, and do not claim success before verification.
```

Listing 2: Feedback-evolution agent context template.

```
{
  "trajectory_to_fix": {
    "score": "<verifier score>",
    "verification_threshold": "<s_verify>",
    "trigger": "<low_score | starvation | drivable_risk |
                collision_risk | hazard_brake_risk |
                oncoming_risk | continuation_risk>",
    "terms": "<per-term verifier subscores>",
    "minimum_progress_m": "<accepted displacement>",
    "minimum_pace_mps": "<execution-pace floor, if set>"
  },
  "trigger_diagnosis": "<measurements behind the trigger>",
  "scene": {
    "images": ["<bird's-eye render>", "<front camera frame>"],
    "summary": "<route, nearby and rear actors, signals>",
    "execution_context": "<remaining route, goal distance,
                          stationary time, signal hold>",
    "continuation": "<check of the planner's next action
                     and of any carried edit>"
  },
  "verifier_feedback": ["<refusal evidence at this state>"],
  "measured_options": ["<option id, speed, pace, scores, verdict>"],
  "attempts_at_this_state": ["<request, resolved controls, verdicts>"],
  "memory": ["<last <M> interventions of this episode>"],
  "final_round_notice": "<only after the wall-time limit>",
  "response_schema": "<fields for each of the five actions>"
}
```

Experience-evolution agent prompt. The experience-evolution agent decides which training tasks are worth practicing next, using the current learner's behavior and the expert's ability to correct it. Listing 3 shows the system prompt and Listing 4 shows the context template.

Listing 3: Experience-evolution agent system prompt.

```
You are the experience-evolution agent for interactive imitation
learning (IIL). Improve what the learner practices by proposing
curriculum changes grounded in learner and expert observations.

DIAGNOSE
- Inspect learner failures, expert corrections, the current
  curriculum, and previous trial outcomes. Cite evidence IDs.
- Identify where the learner needs practice and the expert can
  provide useful corrective supervision.
- A learner-expert score gap suggests an opportunity to learn;
  only a training trial can establish a learner improvement.
- Treat repeated strong learner performance and a small gap to
  a capable expert as evidence for replacing familiar tasks.
```

```
DECIDE
- probe: Request a bounded learner rollout or expert-correction
  probe. State what its result could change about your decision.
- propose: Specify an experience-replacement hypothesis, the
  supporting observations, and a measurable expected change in
  supervision. State what outcome would contradict the hypothesis.
- retain: Keep the accepted curriculum when a change is unsupported.
- defer: Identify missing evidence that prevents a useful decision.

CONSTRAINTS AND MEMORY
- Use only the permitted operations, training pool, and budgets.
- Preserve the curriculum size and respect the replacement limit.
- Keep the learner update, expert-verification rule, evaluation
  protocol, and curriculum-acceptance rule fixed.
- Use previous accepted and rejected trials to refine proposals.
  Distinguish observations, hypotheses, and measured outcomes.
- Return one structured decision in the supplied response schema.
```

Listing 4: Experience-evolution agent context template.

```
{
  "round": "<IIL round>",
  "accepted_learner": "<checkpoint reference>",
  "expert": "<current correction protocol>",
  "curriculum": {
    "tasks": ["<accepted training-task IDs>"],
    "replay": "<fixed replay configuration>"
  },
  "observations": {
    "learner_performance": ["<development evidence>"],
    "expert_feedback": ["<correction and verifier evidence>"],
    "experience_coverage": ["<task visits and learner-expert gaps>"],
    "candidate_experience": ["<training-pool matches>"],
    "probe_results": ["<observations from requested probes>"]
  },
  "memory": ["<prior proposals, outcomes, and evidence references>"],
  "constraints": {
    "allowed_operations": ["replace_experience"],
    "curriculum_size": "<N>",
    "max_replacements": "<K_max>",
    "remaining_probe_budget": "<episodes and calls>",
    "trial_budget": "<collection and learner-update limits>",
    "acceptance_rule": "<fixed learner-evaluation criterion>"
  },
  "response_schema": "<fields required for each decision action>"
}
```

A.2 TOOLS

Feedback-evolution agent tools. Table 2 illustrates the main tools used by feedback-evolution agent in NAVSAFE-∞ simulation environment. Listing 5 illustrates the execution log of an inspection request.

Table 2: Feedback-evolution tools. P: Proposer; O: Orchestrator; V: Verifier. Access names the role that invokes each capability.

Tool	Category	Access	Input and result
`inspect_correction`	Observation	P	Expert trajectory under repair → verifier score and threshold, trigger, per-term subscores, and the progress and pace floors.
`inspect_scene`	Observation	P	Current state → bird's-eye render, front-camera frame, route and actor summaries including rear traffic, signal state, and remaining route.

Continued on next page

Table 2 (continued)

Tool	Category	Access	Input and result
`inspect_continuation`	Observation	P	Planner's next action and any carried edit → a controller-rollout certificate or the reason it failed.
`inspect_history`	Memory	P	Earlier rounds at this state and the last three interventions of the episode → requests, resolved controls, verdicts, and refusal evidence.
`simulate_menu`	Probe	O	Failing state → measured action families with option IDs, speeds, execution pace, scores, and verdicts.
`inspect_edit`	Probe	P	Up to four path, speed, or braking hypotheses → simulated options with the same measurements; nothing executes.
`resolve_clearance`	Probe	P	Actor ID, passing side, and clearance → offsets and window solved from the actor's forecast footprint, returned as simulated options.
`commit_correction`	Decision	P	Menu option or exact edit (offsets at entry, apex, and exit; window; anchor; target speed or acceleration) → an execution request that is checked again before it runs.
`request_stop`	Decision	P	Brake fraction from 0.25 to 1 and a steering reference → a braking correction checked for front and rear contact and road constraints.
`keep_expert_plan`	Decision	P	A refused motion, a complete menu without admissible motion, or a measured actor-geometry refusal → end of the search; the planner's trajectory goes to the final check.
`verify_correction`	Verification	V	Candidate trajectories → teacher-anchored EPDMS score and terms, hard vetoes, and the progress and pace floors.
`certify_execution`	Verification	V	Candidate and live vehicle state → a controller rollout under the recorded actor forecast, with a certificate or a refusal.
`execute_correction`	Execution	O	Admitted action, or the fallback if none is admitted → the vehicle command for the next interval; carried edits are re-certified at later states.
`record_correction`	Collection	O	Executed expert motion → retrospective training windows weighted by the verified score, and a per-replan score log.

Listing 5: Illustrative inspection request comparing a left pass with a gentle stop.

```
{
  "tool_calls": [{
    "name": "set_trajectory_controls",
    "arguments": {
      "label": "compare_left_pass",
      "reason": "Stopped vehicle ahead; the left lane looks clear.",
      "inspect_only": true,
      "hypotheses": [
        {"label": "left_hold",
         "reason": "Pass on the left and hold the offset.",
         "lateral_offset_m": 2.5, "offset_start_m": 3.0,
         "offset_end_m": null, "anchor_route": true,
         "target_speed_mps": 3.0},
        {"label": "gentle_stop",
         "reason": "Fallback if the pass is refused.",
         "brake_to_stop": true, "stop_brake_fraction": 0.5,
         "stop_reference": "planner"}
      ]
```

```
    }
  }]
}
```

Experience-evolution agent tools. Table 3 demonstrates the main tools used by experience-evolution agent in local training environment and we only demonstrate the tools that are related to the IIL cycle. The *probe* tools are applied to learner and expert to obtain learner-expert gap evidence. When feedback-evolution loop is enabled, the latter uses its expert correction procedure. Both return observations before a curriculum trial is launched. Probe requests specify the information needed and the decision that the result could change, as illustrated in Listing 6.

Table 3: Experience-evolution agent tools. P: Proposer; O: Orchestrator; V: Verifier. Access identifies the role using a capability; all execution is managed by the orchestrator.

Tool	Category	Access	Input and result
`inspect_learner`	Observation	P	Learner evaluations → capability performance, failures, and evidence references.
`inspect_feedback`	Observation	P	Correction records → expert quality, learner–expert gaps, and verified supervision.
`inspect_experience`	Observation	P	Current curriculum → task coverage, visits, and evidence of diminishing corrective value.
`inspect_history`	Memory	P	Past trials → hypotheses, accepted or rejected changes, and measured outcomes.
`search_experience`	Acquisition	P	Learning target and task profile → matching training-pool candidates and coverage gaps.
`probe_learner`	Probe	P	Training tasks → learner rollouts with expert alternatives scored at learner-visited states.
`probe_feedback`	Probe	P	Training tasks → expert-takeover outcomes and the availability of verified corrective supervision.
`replace_experience`	Curriculum	P	Acquisition target and replacement count → a proposed fixed-size training curriculum.
`collect_feedback`	Collection	O	Proposed curriculum and learner → learner-visited states, verified expert labels, and collection diagnostics.
`update_learner`	Training	O	Accepted checkpoint and training experience → a candidate learner under the fixed IIL update.
`evaluate_learner`	Evaluation	V	Candidate and accepted learners → performance on the same development tasks and a comparison report.
`commit_curriculum`	State	O	Trial report and acceptance verdict → an updated or retained learner–curriculum pair and a persistent trial record.

Listing 6: Request template for probe tool.

```
{
  "action": "probe",
  "evidence_refs": ["learner_failure_A", "training_task_A"],
  "hypothesis": "Task A may expose a learner weakness that the expert can
    correct.",
  "probe_request": {
    "operation": "probe_feedback",
    "task_ids": ["task_A"],
    "episodes_per_task": 1,
    "information_need": "Can expert takeover produce verified corrective
    supervision?",
    "decision_it_could_change": "Acquire related tasks if corrections are
    available; otherwise revise the target."
  }
```

```
}
```

A.3 AGENTIC LOOPS

The agentic-loop layer is the central design of Agentic DAgger. The harness nests an inner feedback-evolution loop inside learner rollout round and an outer experience-evolution loop around successive learner updates. Each loop owns a different mutable object and *verifier*: feedback-evolution process updates the expert's corrective action at each intervention after local proposal verification, whereas experience-evolution process updates the training curriculum after learner training and development evaluation. The standard DAgger learner update lies between them and is not rewritten by either agent.

A.3.1 FEEDBACK-EVOLUTION LOOP

Inputs and outputs. The feedback context contains the encountered state, learner rollout, existing expert proposals, *verifier* scores and components, and previous attempt diagnostics. Privileged tools expose lane geometry, ego and actor states, traffic controls, scene images, and simulated candidate outcomes. The feedback agent returns a structured action: selecting a measured maneuver, specifying lateral and longitudinal trajectory edits, requesting further inspection, proposing a braking correction, or retaining the planner's trajectory under the permitted conditions. Inspection requests provide additional evidence without executing a correction. For an execution request, the harness constructs the requested trajectory and applies verification and execution checks before returning the admitted corrective action $y^E_{r,t}$. If no correction is admitted, the harness applies its fallback procedure. Previous requests, outcomes, and refusal evidence remain available to guide subsequent decisions.

Configuration and parameter details. PDM-Closed supplies the base expert proposals. The *verifier* uses EPDMS with extended comfort (EC) excluded from the selection objective and history comfort (HC) retained. Expert intervention is triggered when the learner's score falls below 0.7; feedback evolution is invoked when the base expert fails the verification threshold of 0.9. The agent selects a measured maneuver or specifies trajectory edits, which the harness evaluates from the encountered state. The harness checks the agent's selected correction against the verification threshold, hard-safety constraints, and execution requirements; it does not substitute an unchosen higher-scoring alternative. Refusal evidence guides subsequent revisions within the permitted search budget. If no correction is admitted, the harness applies its fallback procedure. The additional time-to-collision intervention trigger is disabled; its configured speed floor of $1\,\mathrm{m/s}$ is therefore inactive.

A.3.2 EXPERIENCE-EVOLUTION LOOP

The loop maintains an accepted learner π^L, a curriculum c, and a trial history $\mathcal{H}$. Its context combines learner evaluations, expert-feedback records, measurements of training experience, and previous trial outcomes. Algorithm 1 describes one curriculum trial cycle, including the decisions that can gather evidence or end the cycle without training. Subsequent cycles reuse the accepted learner–curriculum pair and the expanded history.

Diagnosing the learning value of experience. For each applicable collection episode e, learner and expert scores $q_L(e)$ and $q_E(e)$ define a diagnostic corrective gap $g(e) = \max\{0, q_E(e) - q_L(e)\}$. A large gap is informative only when the expert also satisfies the feedback-quality criterion. It motivates further practice but does not measure the improvement on learner that this round of training will produce. Conversely, a task becomes eligible for mastery-based replacement when repeated observations under the current learner, expert, and collection protocol satisfy

$$q_L(e) \geq \tau_L, \qquad q_E(e) \geq \tau_E, \qquad g(e) < \tau_g, \qquad n \geq n_{\min}, \tag{3}$$

for all applicable episodes, with $n_{\min} \geq 2$ distinct episodes. Measurements from older learner–expert pairs remain historical context rather than satisfying the current replacement criterion.

Proposing a curriculum trial. A proposal specifies a learning target, supporting evidence, an acquisition profile, a replacement count, and a prediction about the supervision to be collected. Listing 7 gives an illustrative decision. For a curriculum of N tasks, the harness constructs $c' =$

$(c \setminus R) \cup A$, where $R \subseteq c$ contains the K removed tasks and A contains K eligible tasks outside c, with $0 < K \leq K_{\max} < N$. Tasks are drawn from the approved training pool. The harness resolves the acquisition profile into concrete task identifiers and checks eligibility and the replacement budget before collection. If no valid allocation exists, it returns the reason for revision. A `retain` decision keeps the curriculum; `defer` records missing evidence.

Listing 7: Illustration of curriculum proposal execution log.

```
{
  "action": "propose",
  "evidence_refs": ["learner_failure_A", "probe_A", "mastery_B"],
  "hypothesis": "Practice on tasks related to A may address the observed
    weakness using available expert corrections.",
  "curriculum_change": {
    "operation": "replace_experience",
    "count": "<K, bounded by K_max>",
    "acquire": {
      "learning_target": "<weak capability identified by
    learner_failure_A>",
      "task_profile": "<training-pool attributes matching that failure>",
      "correction_evidence": "probe_A"
    },
    "retire": {
      "criterion": "Repeated strong learner and expert performance with a
    small corrective gap",
      "evidence_ref": "mastery_B"
    }
  },
  "supervision_prediction": {
    "measure": "<learner-expert gap or verified correction yield>",
    "expected_change": "<measurable change from current experience>"
  },
  "contradicting_result": "The acquired tasks do not provide the predicted
    supervision, or learner performance fails to improve.",
  "regression_risk": "Replacing familiar tasks may reduce retention of
    previously learned behavior."
}
```

Verifying learner improvement. The trial collects expert supervision under c' and trains a candidate from the accepted learner using the fixed IIL update and replay configuration. The *verifier* compares candidate and accepted learners on NAVSAFE-∞. Acceptance requires a valid completed trial and a strict increase in success rate. Target-capability changes and agreement with the supervision prediction are recorded as diagnostic outcomes. They help interpret a trial but do not replace the learner-performance acceptance criterion. In Section 4.3, the full 280-scenario NAVSAFE-∞ suite serves as the development cohort for both diagnosis and selection, and reported scores track optimization on this same target suite.

Memory and a worked interaction. An accepted trial advances both learner and curriculum. A rejected or invalid trial preserves them, while its proposal, realized experience, outcome, and rejection reason enter $\mathcal{H}$. Each subsequent context includes relevant past outcomes, allowing the agent to refine a learning target or avoid repeating an unsupported proposal. Probe results and exhausted attempts are also retained; unavailable measurements are recorded as missing rather than interpreted as learner failures.

Listings 6–8 illustrate this process with execution logs that is processed with anonymization. Suppose a learner failure failure motivates a feedback probe on task A. The probe confirms that expert corrections are available, and repeated observations identify familiar tasks B as eligible for replacement. The agent proposes acquiring tasks related to A. Even if collection supports the supervision prediction, the candidate can fail to improve development success. Listing 8 shows the resulting rejection and memory update. The next decision therefore has evidence about the learning outcome of that acquisition hypothesis, beyond the original learner–expert score gap.

Listing 8: Illustration of execution log where trial outcome returned to experience memory.

Algorithm 1 One experience-evolution trial cycle

Require: Accepted learner π^L, curriculum c, history $\mathcal{H}$, fixed trial configuration $\mathcal{B}$
$\mathcal{C} \leftarrow$ AssembleExperienceContext$(\pi^L, c, \mathcal{H}, \mathcal{B})$
for each permitted decision turn **do**
  $d \leftarrow$ ExperienceAgent$(\mathcal{C})$
  if d is malformed or exceeds the permitted operations or budgets **then**
    Record the error and return repair feedback in $\mathcal{C}$; **continue**
  else if $d.\text{action} = \texttt{probe}$ **then**
    Execute the requested training probe and charge its budget
    Add observations or execution failure to $\mathcal{H}$ and $\mathcal{C}$; **continue**
  else if $d.\text{action} \in \{\texttt{retain}, \texttt{defer}\}$ **then**
    Record d in $\mathcal{H}$; **return** $(\pi^L, c, \mathcal{H})$
  end if
  $c' \leftarrow$ CompileCurriculum$(d, c, \mathcal{B})$
  if the proposed curriculum cannot be constructed **then**
    Record the allocation failure and add it to $\mathcal{C}$; **continue**
  end if
  $D', o \leftarrow$ CollectFeedback$(\pi^L, c', \mathcal{B})$
  if collection satisfies the fixed trial-validity checks **then**
    $\pi' \leftarrow$ IILUpdate$(\pi^L, D', \mathcal{B})$
    Add candidate–parent development evaluation to o
  else
    Mark o as an invalid trial
  end if
  Record d, realized experience, and o in $\mathcal{H}$
  if the trial is valid and development success strictly improves **then**
    return $(\pi', c', \mathcal{H})$
  end if
  return $(\pi^L, c, \mathcal{H})$
end for
Record decision-budget exhaustion; **return** $(\pi^L, c, \mathcal{H})$


```
{
  "trial": "trial_A",
  "collection": {
    "status": "valid",
    "supervision_prediction": "supported"
  },
  "learner_evaluation": {
    "aggregate_success_change": "no improvement"
  },
  "decision": "reject",
  "accepted_state": "learner and curriculum unchanged",
  "memory": {
    "evidence_refs": ["trial_A_collection", "trial_A_evaluation"],
    "finding": "More corrective supervision did not improve the learner
    under this trial budget.",
    "next_context": "Retain the negative result when reconsidering this
    learning target."
  }
}
```


B Experimental Details

B.1 Training and evaluation setup

Training and evaluation. All methods use the same ordered corpus of 100 scenarios from nuPlan Caesar et al. (2021) validation set, which contain 85 log segments across 59 recordings that are

disjoint from the NAVSAFE-∞ scenarios. Each scenario is visited once and lasts up to 15 s. Training is ran with a fixed budget of 100 closed-loop episodes. Each method is trained for 10 cycles where each cycle trains for three round, and we report the best checkpoint performance. For experience-evolution loop, we additionally assign a scenario pool of size 1000 from nuPlan validation split. Every method is evaluated on all 280 NAVSAFE-∞ Bao et al. (2026) scenarios.

Simulation and control. Training uses the simulator environment provided by NAVSAFE-∞. Policies receive front-camera observations and ego-status features, and replan every 0.5 s. Their predicted trajectories are executed by an LQR controller through a kinematic bicycle model running at 10 Hz.

B.2 BASELINE TRAINING CONFIGURATIONS

All methods start from the same pretrained DrivoR Kirby et al. (2026) checkpoint and fine-tune residual decoder adapters while keeping the original model frozen. EnsembleDAgger and ThriftyDAgger use five bootstrapped members and execute their mean trajectory; the other methods use a single actor. We adapt method parameters and policy interfaces to the shared NAVSAFE-∞ driving setup, with method-specific changes identified below.

PDM-Closed Dauner et al. (2023) supplies expert trajectories for Beta-DAgger, EnsembleDAgger, ThriftyDAgger, and AIM. HG-DAgger, PVP, and PPL instead use live human supervision: a person decides when to take and release control and supplies the corrective actions. Labels are stored without teacher-score filtering or trajectory projection.

Supervised actor fitting uses accumulated expert labels with regularization toward the pretrained policy; Beta-DAgger uses stronger proposal anchoring. Episode-based supervised refits are checked against development-set divergence bounds, retaining the latest passing epoch. PVP and PPL use their own learning objectives without these anchors or guards. All methods select its best development iterate.

B.3 ROBOT-SUPERVISED METHODS

Beta-DAgger. Following DAgger Ross et al. (2011), training progressively shifts control from the expert to the novice. Expert labels are collected at every replan regardless of which policy acts, and the actor is refitted on the aggregated dataset after each round.

EnsembleDAgger. Following Menda et al. Menda et al. (2019), expert intervention is triggered by either disagreement between the ensemble mean and the expert or uncertainty across ensemble members. Thresholds are calibrated on initialisation data and then held fixed. Expert labels from every replan are aggregated for ensemble refitting.

ThriftyDAgger. Following Hoque et al. Hoque et al. (2022), ensemble uncertainty estimates novelty, while a learned goal-reaching critic estimates risk. These determine when the expert takes and releases control. The actor learns from expert-controlled states, and autonomous rollouts train the critic within the shared episode budget. Our implementation updates all intervention thresholds each round, including the release-discrepancy threshold held fixed in the paper's algorithm.

AIM. Following AIM Cai et al. (2025a), a learned proxy critic requests expert intervention, and novice–expert agreement determines when control returns to the novice. Actor updates use expert labels during interaction. Our variant initially uses the paper's full-buffer intervention threshold, then switches to a rolling quantile of recent decision-time proxy values.

B.4 HUMAN-SUPERVISED METHODS

Each method collects live human interventions as its novice policy learns. Human takeover decisions determine intervention onset and duration, and human actions provide the expert labels. In our implementation, these actors update between episodes; PVP and PPL updates are therefore batched rather than interleaved with individual environment steps.

HG-DAgger. Following Kelly et al. Kelly et al. (2019), the novice learns to imitate corrective actions collected during human-controlled portions of training. These demonstrations are aggregated, and the actor is refitted after each episode that adds labels. Pretraining replaces demonstration-based initialisation; our single-actor adaptation omits the original ensemble's risk-threshold analysis.

PVP. Following Peng et al. Peng et al. (2023), human corrections and rejected novice actions provide preferred and undesired action signals. A critic learns proxy values from these signals and propagates them through human and novice transitions without environment rewards. The actor learns to maximise the resulting proxy value.

PPL ($L = 0$). Following Cai et al. Cai et al. (2025b), training combines imitation of human corrections with a preference objective favouring those corrections over novice proposals. Our deterministic-actor adaptation uses a reweighted squared-error preference surrogate. Predicted-state preference propagation is omitted because the renderer cannot generate the required observations, so this ablation is reported as PPL ($L = 0$).

C Human Subject Research Protocol

Human Participants. We recruited three volunteer university students (aged 20–30), all of whom held a valid driver's license and reported prior experience with video games. Each participant was briefed in detail on the experimental procedure and provided written informed consent under an IRB-approved protocol. Before the main study, we explained how the agent's predicted trajectories were rendered on screen, and each participant practiced with the control interface in the learning environments. Practice continued until the participant completed ten consecutive successful runs, after which formal data collection began.

Main Experiment. Each participant first completed one or two fully manual episodes to become familiar with the interface. Participants were instructed to intervene only when the agent's predicted trajectory appeared unsafe, illegal, or inconsistent with their intended behavior. They were asked to prioritize safe task completion first, and then to steer the agent toward their personal driving preferences.