HARP: HINDSIGHT REFLECTION AUTONOMOUSLY IMPROVES ROBOT POLICIES

Anonymous authors
Paper under double-blind review

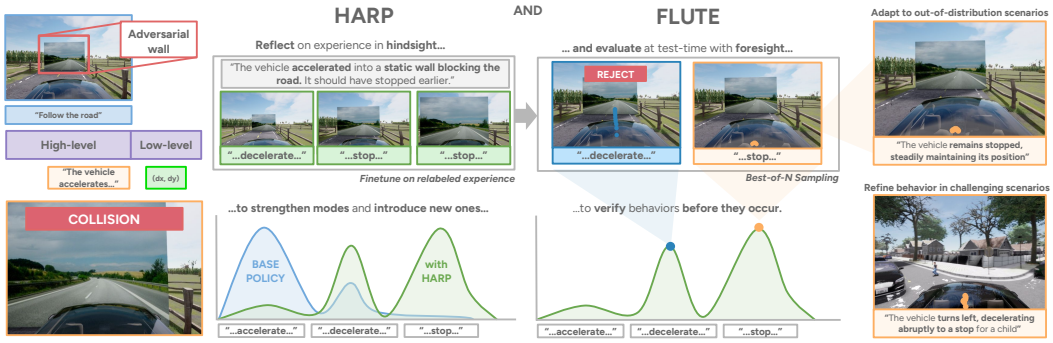


Figure 1: **HARP and FLUTE leverage VLM priors to drive policy improvement.** HARP uses a VLM's common-sense reasoning abilities to introduce and reinforce reasonable actions, and FLUTE leverages a VLM's spatial and semantic priors to evaluate and select proper actions at test time.

ABSTRACT

Real-world robot policies must operate in unseen settings and perform tasks beyond the support of their pretraining data. While scaling up pretraining has enabled broadly capable robot policies, learning autonomously from experience allows policies to continually improve as they encounter new scenarios. A central challenge to achieving this is incorporating prior knowledge into the learning process, allowing robots to interpret their mistakes and determine how to improve their behavior with the benefit of **hindsight**. To address this, we introduce HARP, a general framework for autonomous improvement of language-conditioned robot policies through hindsight reflection. HARP uses the prior knowledge and semantic reasoning capabilities of a vision-language model to reflect on robot experience, determining which decisions led to successful outcomes, caused failures, or introduced risks even when no failure occurred. This feedback is converted into subtask-level corrections, aligned with an overall task strategy. These corrections supervise ongoing updates to a high-level policy, improving its ability to guide the language-conditioned policy through subsequent interactions. To incorporate prior-knowledge guidance at inference time, we apply best-of-N using a VLM-based **foresight** critic, which we term FLUTE, that judges the policy's outputs and selects the most promising candidate. We instantiate HARP and FLUTE for autonomous driving. Combined, they yield a **7.7% improvement** over the base policy in driving performance on a challenging subset of the Bench2Drive benchmark and a **19.4% improvement** on a subset of the Fail2Drive benchmark.

1 INTRODUCTION

Real-world robotic policies require both broad generalization and the ability to improve from new experience. Large-scale pretraining provides a strong foundation for general-purpose capabilities, but adaptation enables robots to refine those capabilities to tackle the unexpected challenges they will inevitably encounter during deployment. Autonomous improvement also allows robot policies to continuously learn from their own experience, reducing the need for additional human demonstrations as robots encounter new tasks and environments. While reinforcement learning (RL) offers a general approach to improvement through autonomous interaction, it typically restricts feedback

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.

to scalar rewards. The RL *policy* can be initialized with prior knowledge about the world through imitation learning, vision-language-action, or world-action model pre-training (Black et al., 2026; Kim et al., 2024; Ye et al., 2026), but the *improvement process* is not able to incorporate more complex feedback taking advantage of this prior knowledge. Scalar reward is inherently a low-bandwidth form of feedback: it cannot explicitly encode *why* a certain strategy was ineffective or what the policy should have done instead, so policies are forced to rely on trial-and-error exploration to learn better behaviors. Not only does this process potentially require substantial interaction, but unsuccessful exploration can result in catastrophic failure.

In contrast, humans make heavy use of prior knowledge not just for their initial attempts at a task, but for structuring the learning and self-improvement process itself, even on nominally successful interactions. For example, a driver who narrowly avoids a collision while driving through a yellow light might revise their strategy to be more attentive at yellow lights in the future, come to a stop earlier, or check for cross-traffic. This behavioral self-refinement is driven by semantics and reflection: detecting a potentially adverse event (the "near-miss"), reflecting on what might have caused it ("I drove through a yellow light"), logically considering alternative courses of action ("I should have stopped"), and then changing the strategy in the future. Much of the knowledge in this process is gained outside of the agent's own driving experience and is imported from other sources.

In this work, we show that pre-trained vision-language models (VLMs) can guide policy improvement in a similar fashion by leveraging their prior knowledge and reasoning capabilities to provide automated semantic feedback. To map the rich language feedback produced by a VLM to low-level control, we rely on hierarchical VLAs (Black et al., 2025; Chen et al., 2026b; Fang et al., 2026; Gao et al., 2026), which decouple decision-making into a high-level semantic policy that predicts intermediate subtasks (such as "pick up the carrot" or "move to the left") and a steerable low-level controller conditioned on those subtasks. This hierarchical structure provides a natural interface for feedback: a standard VLM can express corrections as subtasks, which the low-level policy can convert into grounded actions. This setup effectively leverages both the broad prior knowledge of the VLM and the grounded prior knowledge of the VLA.

We identify two natural points where VLMs can leverage their prior knowledge to refine robotic policies: at training time via hindsight reflection and at inference time via foresight evaluation. We first introduce **H**indsight-guided **A**daptation for **R**obot **P**olicies (**HARP**), a framework for autonomously improving VLA policies through *hindsight reflection*. As shown in Fig. 1, we first collect windows of a policy's behavior. A VLM coach reviews this experience to identify decisions that contributed to undesirable outcomes or created potential risks, and proposes alternative subtask commands. By reviewing experiences in hindsight, the coach can assess decisions in light of their observed consequences. We use these proposed corrections to construct training examples for the high-level policy and incrementally fine-tune the policy, allowing it to improve over the course of interaction. While HARP can introduce new subtasks into a high-level policy's output distribution, it does not guarantee which subtasks will be selected at test-time. To resolve this, we introduce **F**oresight-guided **L**anguage-action **U**tility for **T**est-time **E**valuation (**FLUTE**), the test-time counterpart of HARP. A VLM critic judges subtask proposals and their corresponding actions on what outcomes they *could* lead to. We show that best-of-N rejection sampling with FLUTE can further boost the performance of policies trained with HARP. These two approaches are complementary: by combining HARP and FLUTE, we can ensure the desired behaviors can be generated by the high-level policy and verify these behaviors at test-time.

The main contribution of HARP is a framework that converts VLM hindsight feedback into subtask-level supervision to adapt the policy during interaction. By providing corrections to the high-level policy, HARP leverages prior knowledge to guide a policy to improve from its experience, not only where it failed, but also where it could have done better. To our knowledge, HARP is the first application of hindsight coaching in robotics to autonomously improve VLAs. We instantiate HARP for autonomous driving, where decisions about yielding, passing, and responding to obstacles or other agents early on can significantly affect downstream outcomes. After instilling the high-level policy with the ability to produce strategically beneficial subtasks, we utilize FLUTE, best-of-N with a powerful VLM critic, to evaluate the potential outcomes of proposed actions, ensuring that beneficial adjustments are selected at test time. We evaluate HARP with and without FLUTE on challenging subsets of Bench2Drive (Jia et al., 2024) and Fail2Drive (Gerstenecker et al., 2026), comparing against the base policy and several behavior cloning and online adaptation baselines, finding that HARP and FLUTE in combination achieve 7.7% and 19.4% improvements on chal-

lenging subsets of the Bench2Drive and Fail2Drive benchmarks respectively, matching or greatly exceeding state-of-the-art behavior cloning baselines.

2 Related Work

Language-conditioned and hierarchical robot policies. Language-conditioned policies provide a natural interface for specifying robot behavior, using pretrained visual and language representations or vision-language-action models trained on robot demonstrations (Gadre et al., 2022; Hirose et al., 2024; Allshire et al., 2026; Black et al., 2026; Kim et al., 2024; Renz et al., 2025; Fu et al., 2025; Yang et al., 2026; Ling et al., 2026). These policies borrow the strong perceptual and semantic priors learned during internet-scale pretraining, can be flexibly steered with language prompts, and can generalize across a range of robotic tasks. Hierarchical policies (Black et al., 2025; Gao et al., 2026; Chen et al., 2026b; Fang et al., 2026) decompose task execution into an intermediate reasoning and action prediction step. A high-level policy, often an off-the-shelf or fine-tuned VLM, predicts subtask commands that are used to condition a low-level policy that predicts actions. Prior work has found this decoupling to be advantageous: the low-level policy, which accepts short-horizon commands as input, can focus entirely on execution and control, while the high-level policy handles long-horizon planning and translating this plan into commands.

While these policies have strong capabilities, they are generally trained with large amounts of pre-collected language-labeled robot demonstrations and are therefore limited by their pre-training distributions. With HARP, we build on the subtask interface and devise a method for improving the high-level policy from autonomous experience. A VLM coach observes the policy's behavior *with the benefit of hindsight* and can revise the subtask commands the high-level policy originally predicted. These corrected subtasks are then used as training targets for the high-level policy, autonomously adapting the policy by leveraging its existing hierarchy. Once these corrected subtasks have been instilled into the high-level policy's distribution, FLUTE can leverage a VLM's powerful semantic priors to evaluate the quality of proposed actions. Through best-of-N rejection sampling, FLUTE selects the helpful behaviors introduced by HARP are executed at test-time.

Autonomous improvement for large models. Improving large models with online interaction has become increasingly important to close the gap between pre-training and real-world deployment. In the realm of LLMs, several RL methods (Schulman et al., 2017; Shao et al., 2024; Ouyang et al., 2022) have been developed to address this gap, improving the policy with preference or reward signals. Other works explore "self-refinement" (Madaan et al., 2023; Cheng et al., 2024; Zhang et al., 2025), which uses the same model or a separate critic model to provide feedback on model outputs and iteratively refine them using the in-context learning capabilities of these models. However, both RL and self-refinement methods often make refinements on each model output (rather than over multiple turns) and struggle with improving long-horizon behaviors. In this area, the work most similar to ours is GUI-reflection (Wu et al., 2025), which takes self-refinement a step further by fine-tuning the policy on the corrected actions. However, this method only corrects actions where a failure actually occurred (e.g., the policy clicked on the wrong button). Therefore, mistakes early on that may have led to failure later will not be corrected since those mistakes did not immediately cause the failure. In robotics, addressing early mistakes or risky behaviors is especially important since failure can be irreversible. HARP is designed to address this kind of credit assignment problem through hindsight reflection. As a result, we can refine the *strategy* used by the policy, not just step-level behavior.

Autonomous improvement for robot policies. In the robotics domain, autonomous improvement that integrates prior knowledge has been approached in several ways. Many reinforcement learning methods (Rowe et al., 2025; Zhou et al., 2025; Amin et al., 2025; Zheng et al., 2026; Wagenmaker et al., 2025) start with strong pre-trained policies, most often trained with behavioral cloning, and then use several rounds of RL to improve the policy on a specific task. Some works additionally employ a VLM critic (Lim et al., 2026; Guan et al., 2024; Chen et al., 2026a), either off-the-shelf or with fine-tuning. While a powerful critic can supply external knowledge to the learning process, it still compresses this knowledge into a scalar value to score actions and, moreover, can only provide signal on the actions that the base policy can generate. Other work aims to integrate human knowledge with iterative interventions and finetuning (Liang et al., 2024; Shi et al., 2024; Qi et al., 2026). Yell at Your Robot (Shi et al., 2024) deploys a hierarchical policy and uses human interven-

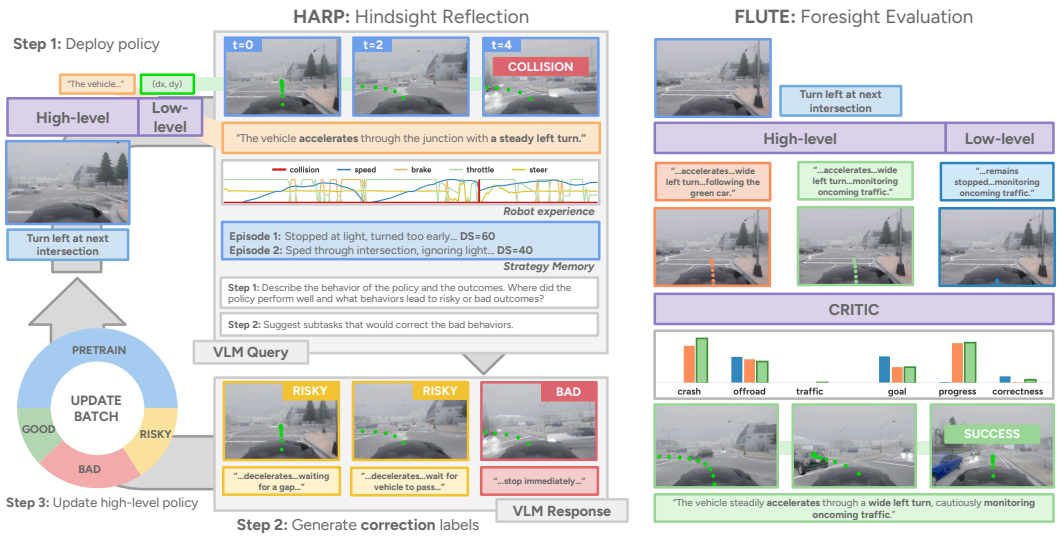


Figure 2: **HARP uses hindsight reflection to autonomously improve policies. FLUTE selects suitable actions at test-time.** We deploy the policy to be improved, collecting experience. Then, we generate correction labels for risky or bad behaviors. The high-level policy is updated on these corrected samples, in addition to samples from pretraining and good behaviors from interaction. We iteratively deploy this updated policy to collect new experience, and repeat the process. At test-time, we deploy policies fine-tuned with HARP with FLUTE, which uses a VLM critic to perform best-of-N sampling and ensure that low-level actions align with high-level intent.

tion to correct the high-level subtask, rolling out the corrected action on-the-fly. Then, the policy is fine-tuned on these corrections, distilling the desired behavior into the policy. ARMOR (Qi et al., 2026) and PHOENIX (Xia et al., 2025) use a mixture of offline and online correction data labeled by humans to learn failure detection and corrections. Alternatively, in-context learning (Fu et al., 2024) has been explored as a way to directly integrate human demonstrations without training.

HARP and FLUTE achieve autonomous improvement without requiring human labeling or intervention. As a result, we can provide dense, natural language feedback on robot policy rollouts in a uniquely scalable fashion.

3 Leveraging VLM Priors to Drive Autonomous Improvement

Our goal is to autonomously adapt a pretrained robot policy to improve its performance in a target scenario through repeated interaction. At time t, the policy receives an observation $\mathbf{o}_t$ and an externally provided task instruction ℓ_t. This instruction typically remains fixed during a task but may change as execution progresses, as with routing commands in autonomous driving. Following prior work (Black et al., 2025; Chen et al., 2026b; Gao et al., 2026), we consider a hierarchical policy consisting of a high-level policy and a language-conditioned low-level policy:

$$\ell_t^{ll} \sim \pi_\theta^{hl}(\cdot \mid \mathbf{o}_t, \ell_t), \qquad \mathbf{a}_t \sim \pi_\phi^{ll}(\cdot \mid \mathbf{o}_t, \ell_t^{ll}).$$

The high-level policy handles common-sense reasoning, planning, and unexpected events that require semantic understanding of images and language. It then outputs a low-level language command ℓ_t^{ll} that guides the low-level policy's selection of appropriate grounded actions $\mathbf{a}_t$. We assume that the low-level policy can reliably execute the types of subtask commands it was trained to follow, which the high-level policy was correspondingly trained to produce.

During interaction, we collect $\mathcal{D}^{\text{exp}}$, consisting of the robot's observations, task instructions, predicted subtasks, and executed actions. From this experience, our objective is to construct a dataset

$$\mathcal{D}^{\text{HARP}} = \{(\mathbf{o}_i, \ell_i, \tilde{\ell^{ll}}_i)\}_{i=1}^N,$$

where each example corresponds to a decision made by the policy during interaction, and $\tilde{\ell^{ll}}_i$ is either a corrected subtask or a retained original command. Our objective is then to improve the high-level policy by generating this dataset and incrementally updating $\pi_\theta^{hl}(\ell^{ll} \mid \mathbf{o}, \ell)$ through supervised

fine-tuning, with the goal of improving task performance on subsequent rollouts. At test-time, our objective is to sample from this fine-tuned policy, with subtasks ℓ_i^{ll} from $\pi_\theta^{hl}(\tilde{\ell^{ll}} \mid \mathbf{o}, \ell)$ and actions $\mathbf{a}_i$ from $\pi_\phi^{ll}(\mathbf{a} \mid \mathbf{o}, \tilde{\ell^{ll}})$ to obtain N language-action candidates $c_i := (\mathbf{a_t}, \ell_t^{ll})_i$, $i \in \{1, \ldots, N\}$. We then aim to evaluate each candidate, selecting the most promising candidate to execute. In the following section, we describe how we generate data with hindsight reflection to fine-tune policies and how we evaluate language-action candidates.

3.1 Improving policies with hindsight reflection and foresight evaluation

We leverage VLMs to facilitate policy improvement on two complementary fronts: hindsight reflection on the policy's interactions with the environment and foresight evaluation of several subtask-action candidates. We first describe HARP, which takes robot experience and produces relabeled subtasks for risky or bad behaviors from a hierarchical policy using the prior knowledge of VLMs. We use this relabeled experience to fine-tune the high-level policy, inducing new, more effective behaviors and strengthening good behaviors already present in the policy. We then describe FLUTE, which uses a VLM critic to inject prior knowledge at test-time to select between the high-level policy's outputs by scoring candidates on their possible future outcomes. In combination, HARP and FLUTE are highly complementary: HARP can change the high-level policy's output distribution, but cannot verify how the low-level policy will respond at test-time, whereas FLUTE is limited by the high-level policy's output distribution but can evaluate its outputs.

Hindsight reflection. To obtain the correction labels, $\mathcal{D}^{\text{HARP}}$, we must first determine where the policy needs to improve. The policy is deployed in the target scenario, collecting experience $\mathcal{D}^{\text{exp}}$ that demonstrates its current behavior. As shown in Fig. 2, the image observations collected by the policy are compiled into a video clip, while the policy inputs and outputs over the window are saved and compiled into a sequence of timestamped metadata. We provide this information to an off-the-shelf VLM, which we use as the source of prior knowledge to reflect on the policy's behavior.

The VLM is first asked to summarize and describe the behavior of the policy at the granularity of each action chunk. It outputs a JSON object that labels each chunk as good, bad, causally bad (could lead to bad behavior) or as having no strong signal, and provides a short description of why that label was given. As shown in Fig. 2, a vehicle might start making a turn which won't directly lead to a bad outcome but may just be risky. Alternatively, once the vehicle has already progressed through the turn, it is immediately clear a collision will occur. Then, the VLM is queried to provide an appropriate correction to the "bad" chunks, in the form of a subtask command, $\tilde{\ell_t^{ll}}$, that *should* have been used to condition the low-level policy. To restrict these corrections to be executable by the low-level policy, we provide examples of in-distribution subtask commands, which guide the VLM to select reasonable subtasks. As a result, we have a dataset $\mathcal{D}^{\text{HARP}} = \mathcal{D}^{\text{corrected}} \cup \mathcal{D}^{\text{correct}}$ where $x_i^{\text{corrected}} \in \mathcal{D}^{\text{corrected}} = (\mathbf{o}_i, \tilde{\ell^{ll}}_i, \ell_i)$, with relabeled subtasks, and $x_j^{\text{correct}} \in \mathcal{D}^{\text{correct}} = (\mathbf{o}_j, \ell_j^{ll}, \ell_j)$, which retains the originally predicted subtasks. While these labels give chunk-level corrections, they need to be unified by a *strategy*. For example, as shown in Fig. 2, a vehicle turning at an intersection may originally speed up to get through a narrow gap, while a less risky strategy would be to wait for a larger gap and edge more cautiously into the turn. Therefore, we also keep a high-level memory of the strategy employed in each episode, summarized as a description of the strategy and the driving score. This memory is provided in the prompt at each call to the VLM for feedback, providing a high-level signal of what strategies have been tried and how they played out.

Updating the high-level policy. With this set of samples, we now have a dataset that can be used to perform supervised fine-tuning of the high-level policy, π^{hl}. However, rather than naïvely updating the policy on batches of samples from $\mathcal{D}^{\text{HARP}}$ alone, we use samples generated from experience as well as the original policy pre-training dataset, $\mathcal{D}^{\text{pretrain}}$. Specifically, we upweight corrective samples by a weight $w_{\text{corrected}}$, as this provides a stronger signal for improving the policy, and weight the other sample pools accordingly. Therefore, the final training mixture is

$$\mathcal{D}^{\text{train}} = w_{\text{corrected}}\,\mathcal{U}(\mathcal{D}^{\text{corrected}}) + w_{\text{correct}}\,\mathcal{U}(\mathcal{D}^{\text{correct}}) + w_{\text{pretrain}}\,\mathcal{U}(\mathcal{D}^{\text{pretrain}})$$

Additionally, we add a small KL penalty to prevent excessive drift from the original policy, $\pi_{\theta_0}^{hl}$. The final objective for finetuning is then

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(\mathbf{o},\ell,\tilde{\ell^{ll}})\sim\mathcal{D}^{\text{train}}}\left[\log \pi_\theta^{hl}(\tilde{\ell^{ll}} \mid \mathbf{o}, \ell)\right]$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{SFT}}(\theta) + \beta\,\mathbb{E}_{(\mathbf{o},\ell)\sim\mathcal{D}^{\text{train}}}\left[D_{\text{KL}}\big(\pi_\theta^{hl}(\cdot \mid \mathbf{o}, \ell) \,\|\, \pi_{\theta_0}^{hl}(\cdot \mid \mathbf{o}, \ell)\big)\right]$$

Foresight evaluation. By fine-tuning the policy with hindsight reflection, we instill it with the ability to generate subtasks that adjust the strategy of the policy towards better behaviors. However, even if the behavior is now included in the high-level policy's output distribution, there is no mechanism to ensure the correct behavior is selected at test-time; we must rely on the high-level policy to predict consistent subtasks. Therefore, we introduce FLUTE, which exploits the generator-verifier gap common to VLMs (Cobbe et al., 2021) to better leverage prior knowledge at test-time. Specifically, FLUTE is a best-of-N method that leverages a VLM critic to evaluate several language-action candidates c_i, consisting of the subtasks sampled from the high-level policy and the corresponding actions predicted by the low-level policy.

For each candidate, the critic is given the policy's current test-time observations ($\mathbf{o_t}$, ℓ_t) together with a visual projection of the action onto the current visual observation. As shown in Fig. 2, we prompt the VLM to independently predict six criteria: expected progress along the route, alignment with ℓ_t, correctness of the proposed action, and whether the proposed action will lead to a collision, a traffic violation, or an off-road violation. For each candidate and criterion, we provide a prompt that grounds the decision in domain-specific knowledge. For each binary criterion, the VLM is prompted to answer `True` or `False`, and we convert the corresponding next-token logits into an approximate probability. Expected progress is similarly computed from the normalized logits over discrete route-advancement choices. We utilize a linear combination of these scores to obtain a reward, and select the maximum-reward option within a safe set:

$$c^* = \arg \max_{c_i \in \mathcal{C}_{\text{safe}}} r(c_i),$$

where $\mathcal{C}_{\text{safe}}$ denotes the set of candidates with risk probabilities each below a set threshold. If all candidates exceed the safety threshold, we select the candidate with the lowest weighted risk. Full details of this calculation, prompts, and example visual observations are provided in Appendix B.

3.2 Implementation

Autonomous driving implementation. To implement HARP and FLUTE for autonomous driving, we adjust our prompts to include some domain-specific guidance that is general across driving tasks. For HARP, we additionally provide examples of the kinds of subtasks that the high-level has been trained to produce and the low-level has been trained to follow, allowing the subtask corrections to be grounded in the support of the policies. The hindsight reflection step has access to the speed of the vehicle, the control inputs, brake, steer, and throttle, and where collisions occurred. We implement HARP with Gemini-3.7-Flash (Google DeepMind, 2026) and FLUTE with Qwen3.8-27B (Qwen Team, 2026). The prompts used for both are provided in appendices B and F.

Training schedule. HARP provides high-level supervision for the high-level policy, and therefore can be deployed in an online setting (e.g., the policy interacts and is updated at the same time) and in an offline setting (e.g., the policy goes through iterative rounds of fine-tuning). We choose to deploy HARP online, updating the policy for 10 updates every 200 environment steps. More details on the hyperparameters used for training are provided in Appendix E.

4 Experimental Results

To evaluate HARP and FLUTE, we explore how they improve policies, independently and in combination, by incorporating prior knowledge from VLMs. Specifically, we aim to answer the following questions:

- **Q1** Do HARP and FLUTE improve performance on challenging and out-of-distribution tasks?
- **Q2** How do HARP and FLUTE compare to state-of-the-art behavioral cloning and autonomous improvement methods?
- **Q3** How do HARP and FLUTE contribute to performance individually and in combination?

4.1 Experimental Setup

Base policies. We use HARP to improve 3 different policies, all trained on the SimLingo (Renz et al., 2025) dataset with language subtasks generated by the refinement pipeline from SteerVLA (Gao et al., 2026). More information on the dataset is provided in Appendix A.1. The

first policy is an existing autonomous driving policy, SteerVLA, which uses an explicitly hierarchical architecture with a separate high-level policy and low-level policy, both fine-tuned from InternVL2-1B (Chen et al., 2024). This policy achieves state-of-the-art performance on both the Bench2Drive and Fail2Drive benchmarks. The second policy, which we call $\pi_{0.5}^{AD}$, is fine-tuned from the $\pi_{0.5}$ (Black et al., 2025) model, which uses a PaliGemma-3B (Beyer et al., 2024) VLM backbone and a flow matching action expert (Amin et al., 2025). The third policy, which we call $\pi_{0.5}^{AD}$ − SteerVLA, uses the SteerVLA high-level policy and the $\pi_{0.5}^{AD}$ low-level policy, allowing us to disentangle the high-level and low-level policy performance. In our experiments, we use HARP to train the high-level component of SteerVLA and use the same frozen low-level policy from $\pi_{0.5}^{AD}$ that is used in the $\pi_{0.5}^{AD}$ experiments.

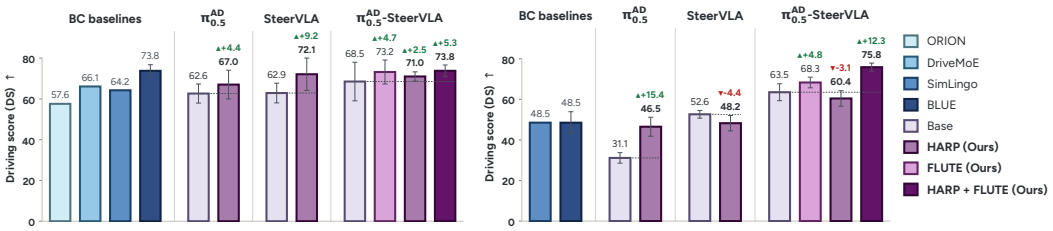
Challenging Bench2Drive and Fail2Drive benchmark subsets. To understand HARP's ability to improve policies on challenging and out-of-distribution tasks, we evaluate HARP on various closed-loop driving scenarios. To target especially challenging routes where there is a possibility for the base policy to improve, we create subsets of the closed-loop autonomous driving benchmarks, Bench2Drive (Jia et al., 2024) and Fail2Drive (Gerstenecker et al., 2026). Bench2Drive evaluates driving abilities in challenging scenarios (e.g., navigating around construction zones) and Fail2Drive evaluates performance in out-of-distribution driving scenarios (e.g., yielding to crossing animals), unseen in pre-training. We use the original metric from these benchmarks to measure driving performance, the driving score (DS), which combines collision avoidance, traffic-rule following, and progress in the route and find routes where several prior behavioral cloning methods struggle. The details of these subset selections are provided in Appendix A. We independently train policies with HARP on each route in the subset, aiming to understand how much we can improve the policy on that particular task. We run the simulator at 20 Hz and collect image observations at a resolution of 1024×512. For each online method, we allow for a budget of 10k environment steps, or about 8 minutes of real-time interaction. If the policy achieves 3 episodes with 100 driving score in a row, we assume the policy has solved the task and finish training early. We then evaluate the policy over 3 evaluation episodes. The raw results are included in Appendix D.

Baselines. We compare the performance of HARP and FLUTE to several behavioral cloning policies: SimLingo (Renz et al., 2025), a chain-of-thought policy trained with counterfactual data to improve generalization, SteerVLA (Gao et al., 2026), a hierarchical policy that uses a fine-grained language interface between a high-level and low-level policy, ORION (Fu et al., 2025), a modular approach that uses long-term history to improve planning, DriveMoE (Yang et al., 2026), a mixture-of-experts method that combines a scene understanding expert and skill expert, and BLUE (Ling et al., 2026), a VLA policy that selectively predicts subtasks rather than predicting language at every step. This set allows us to observe the improvement we can achieve over behavior cloning alone.

We also compare to autonomous improvement methods: residual RL (Johannink et al., 2018) and YAY Robot (Shi et al., 2024). For residual RL (detailed in Appendix C.1), we directly optimize a dense environment reward. For our implementation of YAY Robot, we modify the original framework by replacing the human-in-the-loop with a VLM to act as an automated foresight-intervention mechanism (detailed in Appendix C.2). We provide the VLM with a prompt very similar to the one given to HARP, asking it to determine if a mistake is imminent and to intervene with a correction if so. The high-level policy is then trained on these corrections, which allows us to isolate and determine the specific benefit of *hindsight* in HARP. With these baselines, we observe how conventional reward-based learning compares to the rich feedback we achieve with HARP and FLUTE. We perform this comparison on the Bench2Drive benchmark with SteerVLA only.

4.2 Improving policies on challenging and unseen driving scenarios.

As shown in Fig. 3, HARP achieves improvements over both base policies on the Bench2Drive subset. On the Fail2Drive subset, shown in Fig. 3, HARP achieves a large improvement when applied to the $\pi_{0.5}^{AD}$ policy, while the SteerVLA policy has a slight regression in score. Empirically, we observe that while the SteerVLA high-level policy can produce correct subtasks, the low-level policy is not completely steerable and thus does not follow these corrected subtasks. As a result, the low-level can ignore high-level cues, which results in a slight degradation in behavior. We believe that the discrepancy between the performance improvement on Bench2Drive and Fail2Drive for the SteerVLA policy stems from the types of improvements that are required for each benchmark. While Bench2Drive contains primarily long-tail scenarios, they are still representative of the



(a) Performance on the Bench2Drive subset.

(b) Performance on the Fail2Drive subset.

Figure 3: **Benchmark performance of HARP and FLUTE.** On the selected Bench2Drive and Fail2Drive subsets, HARP + FLUTE improves the hybrid base policy's mean driving score by 5.26 and 12.32 points, respectively, corresponding to relative gains of 7.7% and 19.4%.



(a) With HARP and FLUTE, the policy changes lanes to avoid the cyclists alongside the lane, while the base policy crashes into a cyclist.

(b) With HARP and FLUTE, the policy decelerates and stops before the emergency vehicle, while the base policy accelerates and collides.

Figure 4: **Example rollouts with HARP and FLUTE.** We show examples of HARP and FLUTE with the $\pi_{0.5}^{AD}$ – SteerVLA policy deployed on the `hazard-at-side-lane` route from the Bench2Drive benchmark (left) and the `generalization-fully-blocked` route from the Fail2Drive benchmark (right).

scenarios the policies see in pre-training. Therefore, the autonomous improvement stage mainly targets refinements to timing and strategy rather than introducing entirely new behaviors. While SteerVLA's low-level policy is responsive to these subtask refinements, it struggles to follow the high-level policy in out-of-distribution scenarios.

To combine the stronger base performance from SteerVLA with the steerable low-level behavior from the $\pi_{0.5}^{AD}$ policy, we further investigate HARP and FLUTE on $\pi_{0.5}^{AD}$ – SteerVLA. This policy achieves the strongest base performance on both the benchmark subsets, offering a strong baseline for improvement. For our HARP evaluations, we use the SteerVLA high-level policy fine-tuned with HARP. As shown in Fig. 3, HARP and FLUTE can improve over the base policy by 2.5 points and 4.7 points, respectively. Combined, they improve over the base policy by 5.3 points. Additionally, although HARP alone leads to a small decrease in performance on Fail2Drive, combined with FLUTE, we can improve performance by 12.3 points, far outperforming FLUTE alone.

HARP is critical in far out-of-distribution tasks such as those in the Fail2Drive benchmark. In these cases, the correct behaviors are not in the distribution of the base policy; by utilizing VLM common-sense reasoning to introduce additional reasonable modes for the high-level policy, we are able to produce better actions for FLUTE to select. For example, Fig. 4b depicts a completely out-of-distribution scenario where the vehicle must stop before an emergency vehicle blocking the road. The policy's base distribution only produces accelerating behavior, leading to a collision. HARP reviews the actions leading to the base policy's collision, and introduces additional stopping and decelerating modes to the base policy's distribution. At test-time, FLUTE leverages a VLM's strong visual and semantic reasoning abilities to identify the high collision risk associated with continuing within the lane, leading FLUTE to select a stop as the correct action.

4.3 Comparing HARP to prior methods

With HARP and FLUTE applied to the $\pi_{0.5}^{AD}$ – SteerVLA policy, we match or outperform the strongest baseline despite evaluating on routes deliberately skewed toward scenarios that are particularly challenging for SteerVLA. We match the best method, BLUE (Ling et al., 2026), on Bench2Drive, and outperform the best prior method, SteerVLA, by 23.2 points, on Fail2Drive. We see an especially large gap on Fail2Drive, as HARP and FLUTE are able to leverage prior general-purpose knowledge from VLMs to help the policy generalize to completely unseen scenes.

As an independent autonomous improvement method, HARP with SteerVLA outperforms prior iterative fine-tuning and reinforcement learning methods on the Bench2Drive subset, as shown in Fig. 5. Notably, HARP exhibits a strong boost in performance compared to YAY Robot, demonstrating that hindsight reflection is important for autonomous improvement. We posit that HARP can ground its corrections in actual outcomes, intervening in situations where there are observably bad or risky behaviors, while YAY Robot can only intervene in situations where it *predicts* bad behavior.

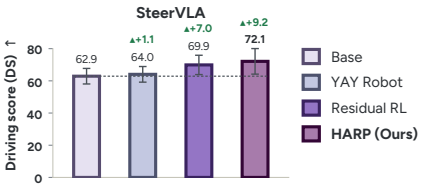


Figure 5: **HARP compared to other autonomous improvement methods.** On the Bench2Drive subset, HARP outperforms prior reinforcement learning and iterative fine-tuning methods.

4.4 Effectiveness of HARP and FLUTE independently and in combination

As shown in Fig. 3a, FLUTE and HARP can independently achieve improvements over base policy performance on the Bench2Drive subset. In combination, we see a performance improvement over either method alone, increasing the average driving score to 73.8. As shown in Fig. 3b, FLUTE alone can improve the average driving score to 68.3, while HARP slightly decreases the score on the Fail2Drive subset. However, their combination results in a 12.3 point improvement to the policy. Alone, FLUTE is limited by the distribution coverage of the base policy. While HARP can introduce new behaviors to the policy, FLUTE can evaluate the actual language-action candidates from the policy at test-time, selecting amongst the various subtask modes of the high-level policy (i.e. "stop" vs. "accelerate") and verifying that the low-level policy produces actions that align with the subtask. As a result, FLUTE and HARP independently perform about the same on the Bench2Drive subset, but, in combination, dramatically improve performance on the Fail2Drive subset.

5 Conclusion

In this work, we proposed HARP, an autonomous improvement method for robot policies that leverages VLMs to perform hindsight reflection, and FLUTE, best-of-N sampling with a novel VLM-based critic to perform test-time evaluation and selection of language-action candidates. We demonstrate that HARP and FLUTE are independently powerful methods for leveraging the prior knowledge of VLMs in the autonomous improvement process and in test-time deployment. HARP utilizes the common-sense reasoning abilities of VLMs to reflect on policy rollouts in hindsight and provide semantic-level corrections to the behaviors of the policy, introducing new modes to the distribution of the high-level policy. FLUTE leverages a VLM's strong spatial and semantic reasoning abilities to identify likely consequences of the actions produced under the guidance of the high-level policy at test-time and selects appropriate language-action candidates to execute. In combination, HARP and FLUTE provide a framework for semantically guided policy improvement and show especially strong improvements on far out-of-distribution tasks, inducing a 19.4% improvement over the base policy on these tasks.

While HARP and FLUTE show strong policy improvements, several avenues remain for future work. In particular, HARP and FLUTE are currently limited by the steerability and action distribution of the low-level policy; a promising direction for future work could be to additionally provide fine-grained adjustments to the low-level policy's output through methods such as residual RL. Overall, we believe that HARP and FLUTE provide a complementary framework for leveraging prior knowledge in *hindsight* in the learning process and in *foresight* to evaluate policy output candidates at test-time.

6 DISCLOSURES

AI USAGE STATEMENT

In accordance with ICLR guidelines, we disclose the use of Large Language Models (LLMs) during the preparation of this manuscript. We utilized LLMs as a general-purpose assistant tool for grammar correction and stylistic editing. All AI-generated text was extensively reviewed and verified. We also used LLMs for assistance with generating plots and figures, which we also extensively reviewed. LLMs did not contribute to the core scientific ideation, methodology, experimental execution, or data analysis, and the authors take full responsibility for the final contents of this paper.

ETHICS STATEMENT

Our work focuses on autonomous improvement for language-conditioned robot policies, specifically evaluated in autonomous driving simulation environments. While our method, HARP, aims to improve the safety and reliability of autonomous vehicles, deploying such models in the real world carries inherent risks. Vision-Language Models (VLMs) can exhibit unpredictable failure modes, hallucinations, or biases inherited from their pre-training data. Consequently, HARP-trained policies should remain in closed-loop simulation environments for rigorous safety validation and should not be deployed in real-world, safety-critical driving scenarios without human oversight, robust safety bounds, and further alignment research. We do not release any datasets containing personally identifiable information.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we have detailed our methodology in Section 3 and provided a comprehensive breakdown of the network architectures, VLM prompts, and hyperparameters in Appendix E. Additionally, we evaluate HARP and FLUTE on standardized, open-source benchmarks (Bench2Drive and Fail2Drive). To facilitate future research, we have included our source code, environment setup instructions, and evaluation scripts in the supplementary materials.

REFERENCES

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, Xi Chen, Rocky Duan, Phillip Isola, Jitendra Malik, Fred Shentu, Guanya Shi, Philipp Wu, and Angjoo Kanazawa. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint*, 2026. URL https://abc.bot/.

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience. 2025. URL https://arxiv.org/abs/2511.14759.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{-}\{0.5\}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, April 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. https://topreward.github.io/webpage/, 2026a. Project page.

William Chen, Jagdeep Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. 2026b.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24185–24198, 2024.

Kanzhi Cheng, Yantao Li, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. Vision-language models can self-improve reasoning via reflection, 2024. URL https://arxiv.org/abs/2411.00855.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL https://arxiv.org/abs/2605.02881.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation, 2025. URL https://arxiv.org/abs/2503.19755.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation, 2022. URL https://arxiv.org/abs/2203.10421.

Tian Gao, Celine Tan, Catherine Glossop, Timothy Gao, Jiankai Sun, Kyle Stachowicz, Shirley Wu, Oier Mees, Dorsa Sadigh, Sergey Levine, and Chelsea Finn. Steervla: Steering vision-language-action models in long-tail driving scenarios, 2026. URL https://arxiv.org/abs/2602.08440.

Simon Gerstenecker, Andreas Geiger, and Katrin Renz. Fail2drive: Benchmarking closed-loop driving generalization, 2026. URL https://arxiv.org/abs/2604.08535.

Google DeepMind. Gemini 3.7 Flash: Model Card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, and Subbarao Kambhampati. Task success is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behaviors, 2024. URL https://arxiv.org/abs/2402.04210.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos, 2024. URL https://arxiv.org/abs/2410.03603.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving, 2024. URL https://arxiv.org/abs/2406.03877.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control, 2018. URL https://arxiv.org/abs/1812.03201.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *8th Annual Conference on Robot Learning*, September 2024.

Jacky Liang, Fei Xia, Wenhao Yu, Andy Zeng, Montserrat Gonzalez Arenas, Maria Attarian, Maria Bauza, Matthew Bennice, Alex Bewley, Adil Dostmohamed, Chuyuan Kelly Fu, Nimrod Gileadi, Marissa Giustina, Keerthana Gopalakrishnan, Leonard Hasenclever, Jan Humplik, Jasmine Hsu, Nikhil Joshi, Ben Jyenis, Chase Kew, Sean Kirmani, Tsang-Wei Edward Lee, Kuang-Huei Lee, Assaf Hurwitz Michaely, Joss Moore, Ken Oslund, Dushyant Rao, Allen Ren, Baruch Tabanpour, Quan Vuong, Ayzaan Wahid, Ted Xiao, Ying Xu, Vincent Zhuang, Peng Xu, Erik Frey, Ken Caluwaerts, Tingnan Zhang, Brian Ichter, Jonathan Tompson, Leila Takayama, Vincent Vanhoucke, Izhak Shafran, Maja Mataric, Dorsa Sadigh, Nicolas Heess, Kanishka Rao, Nik Stewart, Jie Tan, and Carolina Parada. Learning to learn faster from human feedback with language model predictive control, 2024. URL https://arxiv.org/abs/2402.11450.

Jiyu Lim, Youngwoo Yoon, and Kwanghyun Park. The robot's inner critic: Self-refinement of social behaviors through vlm-based replanning, 2026. URL https://arxiv.org/abs/2603.20164.

George Ling, Lijin Yang, Hao Yang, and Zhongzhan Huang. Blue: Toward better language use in efficient vision-language-action models for autonomous driving, 2026. URL https://arxiv.org/abs/2606.08684.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Carl Qi, Xiaojie Wang, Silong Yong, Stephen Sheng, Huitan Mao, Sriram Srinivasan, Manikantan Nambi, Amy Zhang, and Yesh Dattatreya. Self-refining vision language model for robotic failure detection and reasoning, 2026. URL https://arxiv.org/abs/2602.12405.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment, 2025. URL https://arxiv.org/abs/2503.09594.

Luke Rowe, Rodrigue de Schaetzen, Roger Girgis, Christopher Pal, and Liam Paull. Poutine: Vision-language-trajectory pre-training and reinforcement learning post-training enable robust end-to-end autonomous driving, 2025. URL https://arxiv.org/abs/2506.11234.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv preprint arXiv:2403.12910*, 2024.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering, 2025. URL https://arxiv.org/abs/2312.14150.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

Penghao Wu, Shengnan Ma, Bo Wang, Jiaheng Yu, Lewei Lu, and Ziwei Liu. Gui-reflection: Empowering multimodal gui models with self-reflection behavior, 2025. URL https://arxiv.org/abs/2506.08012.

Wenke Xia, Ruoxuan Feng, Dong Wang, and Di Hu. Phoenix: A motion-based self-reflection framework for fine-grained robotic action correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6981–6990, June 2025.

Zhenjie Yang, Yilin Chai, Xiaosong Jia, Qifeng Li, Yuqian Shao, Xuekai Zhu, Haisheng Su, and Junchi Yan. Drivemoe: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving, 2026. URL https://arxiv.org/abs/2505.16278.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Di Zhang, Junxian Li, Jingdi Lei, Xunzhi Wang, Yujie Liu, Zonglin Yang, Jiatong Li, Weida Wang, Suorong Yang, Jianbo Wu, Peng Ye, Wanli Ouyang, and Dongzhan Zhou. Critic-v: Vlm critics help catch vlm errors in multimodal reasoning, 2025. URL https://arxiv.org/abs/2411.18203.

Weicheng Zheng, Yixin Huang, Qiao Sun, Derun Li, and Hang Zhao. Drivema: Driving vision-language-action models with verifiable meta-actions, 2026. URL https://arxiv.org/abs/2605.31271.

Zewei Zhou, Tianhui Cai, Seth Z. Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2506.13757.

Route	ORION	DriveMoE	SimLingo	BLUE	SteerVLA	$\pi_{0.5}^{AD}$	$\pi_{0.5}^{AD}$ − SteerVLA	Avg.
signalized-junction-left-turn-001	18.02	26.92	23.15	**70.33**	60.00	49.03	39.35	40.97
enter-actor-flow-004	60.00	60.00	60.00	56.00	52.00	51.33	**73.43**	58.97
non-signalized-junction-right-turn-003	25.35	60.00	36.00	59.17	**86.67**	78.67	66.33	58.88
non-signalized-junction-left-turn-enter-flow-002	60.00	60.00	48.00	**66.67**	48.00	46.82	62.67	56.02
signalized-junction-left-turn-enter-flow-001	60.00	60.00	60.00	**75.00**	60.00	59.41	19.87	56.33
non-signalized-junction-left-turn-001	16.62	36.00	**100.00**	66.67	73.33	86.67	**100.00**	68.47
signalized-junction-right-turn-004	**60.00**	**60.00**	**60.00**	**60.00**	**60.00**	51.33	**60.00**	58.76
pedestrian-crossing-004	69.49	50.00	80.00	**96.67**	80.00	66.67	66.67	72.79
t_-junction-002	70.00	70.00	70.00	80.00	80.00	**100.00**	79.68	78.53
crossing-bicycle-flow-004	39.96	**100.00**	**100.00**	**100.00**	40.27	60.18	**100.00**	77.20
hazard-at-side-lane-002	**100.00**	**100.00**	60.00	86.67	52.00	78.67	55.93	76.18
merger-into-slow-traffic-v2-005	60.00	97.30	6.40	51.95	45.50	**100.00**	63.67	60.69
vehicle-turning-route-pedestrian-003	**100.00**	17.49	**100.00**	89.33	78.67	24.98	78.67	69.88
highway-exit-002	47.93	**100.00**	60.00	80.10	75.87	31.17	77.88	67.56
accident-two-ways-002	34.97	60.00	**100.00**	66.16	17.26	20.07	55.20	50.52
interurban-actor-flow-004	99.08	**100.00**	64.35	73.23	96.63	97.24	97.24	89.68
Mean	57.59	66.11	64.24	**73.62**	62.89	62.64	68.54	65.09

Table 1: Base performance of BC methods on Bench2Drive subset.

Route	SimLingo	BLUE	SteerVLA	$\pi_{0.5}^{AD}$	$\pi_{0.5}^{AD}$ − SteerVLA	Avg.
generalization-construction-permutations-1019	2.14	9.74	6.94	6.22	**19.40**	8.89
generalization-custom-obstacles-1020	1.85	7.52	10.41	14.73	**18.50**	10.60
generalization-pedestrians-on-road-1085	12.11	28.44	37.81	5.40	**100.00**	36.75
generalization-construction-pedestrian-1011	27.66	37.18	33.06	27.15	**53.51**	35.71
generalization-pedestrian-crowd-1069	38.00	**50.14**	41.68	9.46	23.44	32.54
generalization-custom-obstacles-1024	**58.54**	57.66	46.31	0.29	56.33	43.83
generalization-fully-blocked-1032	33.78	56.56	**59.92**	21.04	49.38	44.14
generalization-hard-brake-1036	59.59	45.14	**76.15**	1.68	47.20	45.95
generalization-pedestrian-other-blocker-1072	47.48	48.03	48.22	**66.67**	**50.00**	48.75
generalization-right-construction-1094	67.46	50.61	91.87	78.23	**100.00**	77.63
generalization-right-of-way-1056	**100.00**	60.00	65.27	36.00	**100.00**	72.25
generalization-image-on-object-1041	**93.95**	40.92	93.05	27.85	58.72	62.90
generalization-obscured-stop-1046	**100.00**	80.58	95.41	69.33	**100.00**	89.06
generalization-bad-parking-1004	58.03	52.53	45.78	30.00	**100.00**	57.27
generalization-animals-1076	10.56	**77.44**	24.99	58.10	75.00	49.22
generalization-wall-1097	65.00	**73.17**	65.00	20.01	65.00	57.64
Mean	48.51	48.48	52.62	31.12	**63.53**	48.32

Table 2: Base performance of BC methods on Fail2Drive subset.

A Fail2Drive and Bench2Drive benchmark subsets

We compare 5 prior behavioral cloning baselines on these full benchmarks: BLUE (Ling et al., 2026), SimLingo (Renz et al., 2025), ORION (Fu et al., 2025), DriveMoE (Yang et al., 2026) and SteerVLA (Gao et al., 2026), finding routes where all policies perform poorly. As we use SteerVLA as one of our base policies, we specifically exclude routes where it gets a driving score of 100, to avoid biasing towards the base policy performance. We then filter these routes to a diverse set (e.g., we keep one route that tests signalized right turns rather than all five) and those that do not require camera views beyond the front camera (e.g., we exclude routes that require yielding to an emergency vehicle approaching from behind), since, in these cases, the only way to improve the policy is to unrealistically memorize an unobservable yet critical event. We do this for Bench2Drive and Fail2Drive, selecting a subset of 16 routes from each. In Table 1 and Table 2, we provide the per route scores of the various behavior cloning baselines as well as the base policies we use in our experiments. We include the full evaluation data for SteerVLA and $\pi_{0.5}^{AD}$ in our supplemental submission. The other BC baselines have their seeds available through their original papers.

A.1 Base Policies

We apply HARP and FLUTE to three different base policies. SteerVLA (Gao et al., 2026) is a hierarchical policy consisting of a high-level VLM that guides a low-level VLA, both fine-tuned from (Chen et al., 2024), with a fine-grained meta-action interface, making it inherently compatible

with HARP. The $\pi_{0.5}^{AD}$ policy is fine-tuned from $\pi_{0.5}$ (Black et al., 2026), and utilizes PaliGemma3B as its backbone with a language-conditioned flow-matching expert. $\pi_{0.5}^{AD}$ is trained using the same language labels as SteerVLA, making the two policies compatible; thus, we additionally utilize $\pi_{0.5}^{AD}$ – SteerVLA, which uses the SteerVLA high-level policy to condition the $\pi_{0.5}^{AD}$ flow-matching action expert.

All base policies are trained on the SimLingo (Renz et al., 2025) dataset with language labels refined by the SteerVLA labeling pipeline. This consists of a set of 3.1 million samples collected at 4 Hz by deploying the PDM-Lite CARLA driving expert (Sima et al., 2025) on a set of routes in CARLA Towns 1-10, 12, and 13, each with various driving scenarios (e.g., merging around an accident in the lane). None of the base policies are trained on data from the Fail2Drive scenarios.

Method	B2D DS	F2D DS
SteerVLA	91.20 ± 0.91	81.30 ± 0.69
BLUE	90.51 ± 0.37	79.77 ± 0.71
SimLingo	86.56	76.76
ORION	77.74	—
DriveMoE	74.22	—
Base Policy	73.76 ± 0.59	60.50 ± 0.72

Table 3: **Full benchmark performance of SoTA BC methods and base policies.**

B FLUTE Details

To prompt FLUTE to first ground its judgment in the visual components of the scene, we first provide the vehicle's visual observation as a 704×352 image. We ask the VLM to describe the scene using the prompts in Listings 1 and 2. We then provide general driving-domain and input format context to the model with the template shown in Listing 3. We prompt the model to respond with a specific answer choice using the criterion-specific addenda in Listings 4 to 10. We then utilize the VLM's underlying probabilistic distribution to obtain a reward; details of this calculation are specified in Appendix B.1. Since the obstacles encountered in the Fail2Drive benchmark are much more out-of-distribution, the base policy is less cautious, and therefore produces actions that are more likely to lead to collisions. Thus, we slightly amend the prompt provided to the critic to determine the collision probability, up-weight the collision penalty in our utility calculation, and reword the description prompt to be broader.

Listing 1: Bench2Drive First Stage Prompt.

```
Describe the visible driving scene in two concise sentences: the road and
    lane layout, road edges, nearby road users or obstacles and their
   positions, and any visible traffic controls relevant to the ego
   vehicle. Distinguish what is visible from what is uncertain; do not
   invent hidden signals or prior stopping behavior.
```

Listing 2: Fail2Drive First Stage Prompt.

```
Describe the objects ahead and their positions relative to the vehicle's
   lane in one sentence.
```

Listing 3: Second stage general driving context.

```
This is the CARLA driving simulator. Evaluate the complete 10-step action
    over 2.5 seconds, even if the controller will execute only a shorter
    prefix before querying again. The translucent rectangles are added
   future VEHICLE FOOTPRINT graphics; their connecting line and numbered
    centers describe the planned path. Cyan/blue to pink means earlier
   to later time, not safety. The widths represent vehicle width. Image
   overlap alone is not physical contact or a depth measurement. A
   stopped action may have no visible footprint because it stays at the
   current camera-relative position. Off-screen or unsupported
   footprints are unknown, not evidence of safety.
```

```
Scene description (verify against image; may be wrong): {generated
    description}
Routing command: {routing command}
Candidate intent: {subtask}
Combined trajectory: {timed positions and travel distances}
Full temporal XY positions: {ten positions}
Full spatial XY path: {ten positions}
{Criterion-specific instruction}
```

Listing 4: Off-road violation identification prompt.

```
Predict the outside-route-lanes penalty during movement over this 2.5-
    second action. The geometric test uses the VEHICLE CENTER, not tire
    or footprint corners: it is outside road when its distance to the
    nearest driving or parking lane centerline exceeds half that lane
    width plus 0.5 meters. Driving and parking lanes count as road;
    sidewalks, grass and non-driving shoulders do not become lanes merely
     because they are flat or paved. Crossing a painted divider into
    another same-direction lane, or turning within a junction, is not by
    itself off-road. The criterion also counts travel against a lane
    direction when its wrong-direction check is enabled, but suppresses
    that direction check inside junctions and in scenarios that
    explicitly disable it. Do not infer such an exemption solely from the
     candidate intent. The penalty accumulates distance traveled outside
    these allowed lanes; this question asks whether any such travel
    occurs, not its eventual route-wide percentage. A stopped candidate
    adds no distance. Use visible lane geometry to estimate the test;
    hidden map boundaries and scenario switches are not supplied and must
     not be invented.

Statement: This action adds distance penalized as outside route lanes.
Answer with exactly True or False. Answer:
```

Listing 5: Traffic violation identification prompt.

```
Predict an actual red-light or stop-sign infraction within these 2.5
    seconds. A red-light infraction requires crossing the stop line of a
    RED signal controlling this lane and movement; approaching or
    stopping before it, a signal for other traffic, or being in a
    junction after entering legally is not sufficient. A stop-sign
    infraction requires passing through the controlling stop region
    without the required full stop (speed below 0.1 m/s while affected by
     that same sign). A stop already completed for that encounter
    satisfies it. The image does not reveal all stopping history, signal
    timing or trigger geometry: do not assume either a prior stop or a
    violation from the sign alone. Yielding to vehicles, lane choice and
    speeding are separate issues, not this term.

Statement: This action incurs a red-light or stop-sign infraction.
Answer with exactly True or False. Answer:
```

Listing 6: Routing command compatibility prompt.

```
Judge ROUTE COMPATIBILITY of the proposed action: whether its spatial
    path takes or preserves the next maneuver required by the routing
    command, so the instructed route remains achievable. It need not
    finish the entire route within 2.5 seconds. A path toward a wrong
    branch, away from the required turn, or into an unrecoverable
    position is incompatible. Staying stopped on the intended approach
    can remain route-compatible. Do not reward distance or penalize
    traffic/collision risk again here; those are separate terms. The
    candidate intent is a claim; verify it against its path and the scene
    .
```

```
Statement: This action is compatible with completing the instructed route
    .
Answer with exactly True or False. Answer:
```

Listing 7: Action correctness prediction prompt.

```
Judge whether an attentive competent driver following the instructed
    route would consider this complete steering AND timing plan an
    appropriate immediate maneuver. Consider right of way, available gaps
    , predictable interactions, needed yielding and whether waiting or
    moving serves the current situation. Several different actions can be
     appropriate; exact imitation of one hidden expert trajectory is not
    required. Safe but unnecessary indefinite stopping on a clear
    approach is not automatically appropriate. Candidate intent alone
    does not establish correctness. No actual expert plan is provided to
    this query.

Statement: This is an appropriate immediate driving maneuver for this
    situation.
Answer with exactly True or False. Answer:
```

Listing 8: Progress prediction prompt.

```
Estimate net ADVANCEMENT ALONG THE INSTRUCTED ROUTE during the next 2.5
    seconds, to the nearest meter. Use the temporal head for distance
    traveled and the spatial head for path shape; do not count the two
    heads twice. Following a correct curved turn can advance the route
    even if its straight-line displacement is small. Motion down a wrong
    branch, sideways away from the route, or backward is not positive
    route advancement. A stationary action is zero. Do not separately
    discount predicted contact or traffic violations here: score the
    proposed geometric advancement, with risk handled by the other terms.
     Clamp negative advancement to zero and 10 meters or more to 10. If
    route geometry is ambiguous, use the visible road and routing
    instruction rather than pretending to know an unseen route map.

Distance choices: A: 0 m; B: 1 m; C: 2 m; D: 3 m; E: 4 m; F: 5 m; G: 6 m;
     H: 7 m; I: 8 m; J: 9 m; K: 10 m or more
Answer with exactly one letter A through K. Answer:
```

Listing 9: Bench2Drive-specific collision prompt.

```
Predict physical contact during the complete next 2.5 seconds. Compare
    the full vehicle footprint with cars, pedestrians, cyclists, barriers
     and other solid objects, including their plausible motion and
    crossing conflicts. A small clearance or nearby object alone is not a
     collision. A stopped vehicle can still be struck by another road
    user. Use distance and timing, not candidate intent, as evidence.

Statement: Physical contact will occur during this action.
Answer with exactly True or False. Answer:
```

Listing 10: Fail2Drive-specific collision prompt.

```
Judge physical contact during the next 2.5 seconds covered by the
    complete proposed action. Consider the surrounding road, obstacles,
    and how nearby vehicles and people may move during that interval. Use
     the footprint positions together with their times; proximity by
    itself does not establish contact. Do not extend the prediction
    beyond the action window. Closely inspect objects occupying or moving
     into the cyan/blue and pink/magenta projected vehicle footprints,
    including the swept path between footprints. During a lane change,
    check vehicles alongside and in the destination lane, including their
     predicted motion and closing distance. The colors indicate earlier
    and later times along one candidate path, not separate safe and
```

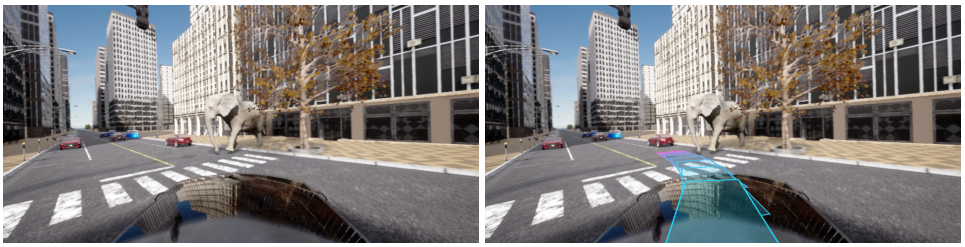


Figure 6: **Example visual observations provided to FLUTE**. In the description stage, the VLM is provided the raw scene observation and prompted to generate a short scene description; this description is shared across candidates, and used to condition logit generation for criteria scores in the visual observation. In the second stage, the candidate action $\mathbf{a_{ti}}$ is projected onto the image to provide visual grounding for the proposed action.

```
    unsafe paths. Use perspective, numerical positions and timing to
    distinguish physical overlap from an image-only overlap; do not
    invent unseen vehicles.

Statement: The proposed driving trajectory will avoid physical collisions
    .
Answer with exactly True or False. Answer:
```

B.1 FLUTE Reward Calculation

For each criterion described in Listings 4 to 7, 9 and 10, we read the VLM's next-token logits for `True` and `False` and normalize over these two answers:

$$p_j(c_i) = \frac{\exp z_j^{\texttt{True}}(c_i)}{\exp z_j^{\texttt{True}}(c_i) + \exp z_j^{\texttt{False}}(c_i)}.$$

For progress, we instead normalize the logits of eleven answer choices representing $0, \ldots, 10$ meters of route advancement and take their expected value:

$$\mathbb{E}[\Delta_{\text{progress}}(c_i)] = \sum_{m=0}^{10} m\, p(m \mid \mathbf{o}_t, \ell_t, c_i).$$

We define our reward to be:

$$r(\mathbf{o_t}, \ell_t, c_i) = \lambda_{\text{progress}}\mathbb{E}[\Delta_{progress}(c_i)] + \lambda_{\text{goal}}p_{\text{goal}}(c_i) + \lambda_{\text{correct}}p_{\text{correct}}(c_i)$$
$$-\lambda_{\text{crash}}p_{\text{crash}}(c_i) - \lambda_{\text{traffic}}p_{\text{traffic}}(c_i) - \lambda_{\text{offroad}}p_{\text{offroad}}(c_i),$$

where λ_j denotes a pre-defined coefficient for each term. We select

$$c^* = \arg\max_{c_i \in \mathcal{C}_{\text{safe}}} r(c_i),$$

$$\mathcal{C}_{\text{safe}} = \{c_i \mid p_{\text{crash}}(c_i) < \zeta_{\text{crash}},\ p_{\text{traffic}}(c_i) < \zeta_{\text{traffic}},\ p_{\text{offroad}}(c_i) < \zeta_{\text{offroad}}\},$$

where ζ_j is a set threshold. Coefficients and thresholds are provided in Table 14.

C Baselines

C.1 Residual Reinforcement Learning

We adapt a frozen SteerVLA policy with residual Soft Actor-Critic (SAC) (Haarnoja et al., 2018). At each control step, the frozen base policy produces a waypoint chunk, which is decoded by the CARLA PID controller into a base acceleration–steering action $a_t^{\text{base}} \in [-1, 1]^2$. The residual

SAC actor takes the frozen SigLIP state embedding and a_t^{base} as input, and samples an unscaled, tanh-squashed residual $\delta_t \in [-1, 1]^2$. The executed action is

$$a_t = \text{clip}\big(a_t^{\text{base}} + s_t \odot \delta_t,\ -1, 1\big)\,,$$

where $s_t = [s_t^{\text{acc}}, s_t^{\text{steer}}]$ is the residual-scale vector. We execute the base policy alone during an initial warmup period, then linearly ramp residual authority from zero to its final scale.

We use the standard maximum-entropy SAC objective with twin critics, a learned temperature, target critics, and Polyak target updates. The actor and each critic are MLPs with layer normalization, and the target entropy is set automatically. To discourage unnecessarily large deviations from the base policy, we add an unnormalized L2 penalty on the *scaled* residual, with coefficient β_{BC}. All hyperparameters are listed in Table 15.

The residual state is the concatenation of frozen, L2-normalized SigLIP image and route-prompt embeddings. The base SteerVLA policy remains frozen throughout residual training; it queries the low-level policy every three environment steps and refreshes its high-level reasoning every five low-level queries. Each training run is followed by frozen-policy evaluation episodes.

Reward. Let Δp_t be nonnegative route-progress percentage accrued at step t, v_t the ego speed, $v_t^{\max}$ the local speed limit, c_t an indicator that a new collision occurred or collision contact remains active, Δo_t the outside-route infraction increment, and $\Delta \ell_t$ the traffic-violation increment. The reward is

$$r_t = 0.05\,\Delta p_t + b_t - 0.1\max\left(0, \frac{v_t}{v_t^{\max}} - 1\right) - 20\,c_t - 20\,\Delta o_t - 20\,\Delta \ell_t$$

Here, crash–stuck means that the vehicle remains below 0.1 m/s for 20 simulator steps after a counted collision. The terminal term b_t is applied once at the end of the episode: $b_t = 50$ on successful route completion, $b_t = -5$ upon crash–stuck termination (when the vehicle remains below 0.1 m/s for 20 simulator steps following a collision), $b_t = 0$ for max-step horizon terminations, and $b_t = -20$ for all other failure terminations.. Thus, the reward primarily measures route progress while penalizing overspeeding, safety events, lane departures, and traffic violations.

C.2 Yell at Your Robot

We implement YAY Robot (Shi et al., 2024) as a foresight language-correction baseline on a hierarchical SteerVLA policy. The frozen low-level controller converts a high-level reasoning trace and subtask into driving actions. Before each newly generated high-level chain-of-thought (CoT) is passed to the low-level controller, a VLM receives the current forward-facing image, route command, ego telemetry, recent subtask history, and the policy's proposed reasoning and subtask. The VLM either retains the proposal or supplies a corrected reasoning–subtask pair. When a correction is issued, it immediately conditions the low-level controller for the current action query; unlike hindsight relabeling, it therefore changes behavior online.

For SteerVLA, the low-level policy is queried every three environment steps and the high-level CoT is refreshed approximately every six environment steps. We query the correction model every four CoT refreshes, or approximately every 24 environment steps. We retain an eight-entry episode-level strategy memory. This differs from the original human-feedback setting of YAY Robot: we use a VLM as the correction source while preserving the central foresight-intervention mechanism.

Each corrected frame is stored as a direct corrective example, labeled `BAD`. We additionally label the preceding two seconds of observations (40 simulator steps) with the same corrected language as `BAD` precursor examples. If the VLM retains the original CoT, we store the frame as a `GOOD` example. Each online high-level update batch draws 50% of its examples from an offline SteerVLA high-level dataset and 50% from the accumulated online correction dataset. Within the online portion, 90% of examples are corrective, and two-thirds of those are precursor examples.

Only the high-level policy is updated; the low-level waypoint policy is frozen and no residual-RL or low-level behavior-cloning updates are performed.We perform 10 high-level gradient steps every 200 environment steps. After training, we freeze the adapted policy and evaluate it. All hyperparameters are listed in Table 16.

Table 4: **HARP results for the $\pi_{0.5}^{AD}$ and SteerVLA policies on the Bench2Drive subset.** We evaluate HARP with the $\pi_{0.5}^{AD}$ and SteerVLA policies on a challenging 16-route subset from Bench2Drive. Results are shown over 3 evaluation seeds.

	$\pi_{0.5}^{AD}$			SteerVLA		
Route	Base	HARP	Δ	Base	HARP	Δ
signalized-junction-left-turn-001	49.03	56.19	+7.16	60.00	71.71	+11.71
enter-actor-flow-004	51.33	86.67	+35.34	52.00	65.33	+13.33
non-signalized-junction-right-turn-003	78.67	60.00	-18.67	86.67	86.67	+0.00
non-signalized-junction-left-turn-enter-flow-002	46.82	67.53	+20.71	48.00	71.07	+23.07
signalized-junction-left-turn-enter-flow-001	59.41	73.33	+13.92	60.00	60.00	+0.00
non-signalized-junction-left-turn-001	86.67	45.00	-41.67	73.33	100.00	+26.67
signalized-junction-right-turn-004	51.33	67.33	+16.00	60.00	100.00	+40.00
pedestrian-crossing-004	66.67	62.00	-4.67	80.00	43.33	-36.67
t_-junction-002	100.00	72.05	-27.95	80.00	85.33	+5.33
crossing-bicycle-flow-004	60.18	52.13	-8.05	40.27	100.00	+59.73
hazard-at-side-lane-002	78.67	65.33	-13.34	52.00	86.67	+34.67
merger-into-slow-traffic-v2-005	100.00	100.00	+0.00	45.50	45.72	+0.22
vehicle-turning-route-pedestrian-003	24.98	69.58	+44.60	78.67	68.76	-9.91
highway-exit-002	31.17	78.62	+47.45	75.87	32.64	-43.23
accident-two-ways-002	20.07	17.71	-2.36	17.26	39.14	+21.88
interurban-actor-flow-004	97.24	99.08	+1.84	96.63	97.33	+0.70
Mean ± σ	62.64 ± 4.68	67.03 ± 7.05	+4.39	62.89 ± 4.83	72.11 ± 7.94	+9.22

D Extended Results.

We show the full results per route over all methods. The HARP results for the $\pi_{0.5}^{AD}$ and SteerVLA policies are shown in Table 6 and Table 7 show the full results for the $\pi_{0.5}^{AD}$ − SteerVLA policy on the Bench2Drive and Fail2Drive subsets. The Residual RL and YAY Robot results for the SteerVLA policy on the Bench2Drive subset are shown in Table 8 and Table 9.

D.1 Main results

Table 5: **HARP results for the $\pi_{0.5}^{AD}$ and SteerVLA policies on the Fail2Drive subset.** We evaluate HARP with the $\pi_{0.5}^{AD}$ and SteerVLA policies on a challenging 16-route subset from Fail2drive. Results are shown over 3 evaluation seeds.

	$\pi_{0.5}^{\text{AD}}$			SteerVLA		
Route	Base	+ HARP	Δ	Base	+ HARP	Δ
construction-permutations-1019	6.74	3.42	−3.32	6.94	21.05	+14.11
custom-obstacles-1020	14.31	18.23	+3.92	10.41	20.30	+9.89
pedestrians-on-road-1085	5.40	30.83	+25.43	37.81	36.16	−1.65
construction-pedestrian-1011	28.10	43.33	+15.23	33.06	34.40	+1.34
pedestrian-crowd-1069	8.71	9.99	+1.28	41.68	35.14	−6.54
custom-obstacles-1024	0.31	46.67	+46.36	46.31	30.68	−15.63
fully-blocked-1032	26.03	60.00	+33.97	59.92	60.00	+0.08
hard-brake-1036	1.68	25.94	+24.26	76.15	52.00	−24.15
pedestrian-other-blocker-1072	66.67	100.00	+33.33	48.22	83.33	+35.11
right-construction-1094	78.23	73.33	−4.90	91.87	32.75	−59.12
right-of-way-1056	40.45	38.48	−1.97	65.27	51.90	−13.37
image-on-object-1041	31.62	78.53	+46.91	93.05	100.00	+6.95
obscured-stop-1046	69.33	20.50	−48.83	95.41	100.00	+4.59
bad-parking-1004	27.20	52.00	+24.80	45.78	44.00	−1.78
animals-1076	58.10	42.79	−15.31	24.99	41.80	+16.81
wall-1097	35.01	100.00	+64.99	65.00	28.30	−36.70
Mean DS ↑	31.12 ± 2.58	46.50 ± 4.68	+15.39	52.62 ± 1.90	48.24 ± 3.77	−4.38

D.1.1 Base policy comparison

Table 6: **HARP + FLUTE results on the Bench2Drive subset with $\pi_{0.5}^{AD}-\mathrm{SteerVLA}$.** We evaluate HARP and FLUTE on a challenging 16-route subset from Bench2Drive. Results are shown over 3 evaluation seeds.

Route	Base	HARP	Δ	FLUTE	Δ	HARP + FLUTE	Δ
signalized-junction-left-turn-001	39.35	62.36	+23.01	73.33	+33.99	54.51	+15.17
enter-actor-flow-004	73.43	60.00	-13.43	73.33	-0.10	73.33	-0.10
non-signalized-junction-right-turn-003	66.33	76.31	+9.98	63.67	-2.67	75.00	+8.67
non-signalized-junction-left-turn-enter-flow-002	62.67	37.29	-25.38	27.99	-34.68	60.00	-2.67
signalized-junction-left-turn-enter-flow-001	19.87	50.23	+30.36	72.74	+52.87	49.98	+30.10
non-signalized-junction-left-turn-001	100.00	73.33	-26.67	86.67	-13.33	73.33	-26.67
signalized-junction-right-turn-004	60.00	84.79	+24.79	57.33	-2.67	86.67	+26.67
pedestrian-crossing-004	66.67	86.67	+19.99	80.00	+13.33	34.68	-31.99
t_-junction-002	79.68	69.77	-9.91	100.00	+20.32	74.87	-4.81
crossing-bicycle-flow-004	100.00	100.00	+0.00	100.00	+0.00	100.00	+0.00
hazard-at-lane-side-002	55.93	86.67	+30.74	100.00	+44.07	100.00	+44.07
merger-into-slow-traffic-v2-005	63.67	81.83	+18.17	63.89	+0.23	100.00	+36.33
vehicle-turning-route-pedestrian-003	78.67	62.85	-15.81	71.67	-7.00	78.67	+0.00
highway-exit-002	77.88	77.88	+0.00	77.88	+0.00	76.51	-1.37
accident-two-ways-002	55.20	29.87	-25.33	25.95	-29.25	46.87	-8.33
interurban-actor-flow-004	97.24	96.33	-0.91	96.94	-0.30	96.33	-0.91
Mean $\pm$ σ	68.54 $\pm$ 9.46	71.01 $\pm$ 2.28	+2.47	73.21 $\pm$ 5.89	+4.68	73.80 $\pm$ 2.85	+5.26

Table 7: **HARP + FLUTE results on Fail2Drive subset with $\pi_{0.5}^{AD}-\mathrm{SteerVLA}$.** We evaluate HARP and FLUTE on a challenging 16-route subset from Fail2Drive. Results are shown over 3 evaluation seeds.

Route	Base	HARP	Δ	FLUTE	Δ	HARP + FLUTE	Δ
generalization-construction-permutations-1019	19.40	17.85	-1.55	27.10	+7.71	33.83	+14.44
generalization-custom-obstacles-1020	18.50	11.64	-6.86	13.31	-5.19	9.82	-8.68
generalization-pedestrians-on-road-1085	100.00	100.00	+0.00	100.00	+0.00	100.00	+0.00
generalization-construction-pedestrian-1011	53.51	61.75	+8.24	68.86	+15.35	52.95	-0.56
generalization-pedestrian-crowd-1069	23.44	23.51	+0.07	39.01	+15.57	39.01	+15.57
generalization-custom-obstacles-1024	56.33	65.00	+8.67	65.00	+8.67	55.54	-0.79
generalization-fully-blocked-1032	49.38	88.33	+38.95	59.94	+10.56	100.00	+50.62
generalization-hard-brake-1036	47.20	86.67	+39.47	100.00	+52.80	100.00	+52.80
generalization-pedestrian-other-blocker-1072	50.00	50.00	+0.00	50.00	+0.00	83.33	+33.33
generalization-right-construction-1094	100.00	100.00	+0.00	69.50	-30.50	86.67	-13.33
generalization-right-of-way-1056	100.00	69.78	-30.22	100.00	+0.00	100.00	+0.00
generalization-image-on-object-1041	58.72	41.22	-17.50	38.08	-20.64	100.00	+41.28
generalization-obscured-stop-1046	100.00	100.00	+0.00	100.00	+0.00	100.00	+0.00
generalization-bad-parking-1004	100.00	73.33	-26.67	100.00	+0.00	86.67	-13.33
generalization-animals-1076	75.00	12.93	-62.07	96.67	+21.67	77.41	+2.41
generalization-wall-1097	65.00	65.00	+0.00	65.00	+0.00	88.33	+23.33
Mean $\pm$ σ	63.53 $\pm$ 4.17	60.44 $\pm$ 3.83	-3.09	68.28 $\pm$ 2.62	+4.75	75.85 $\pm$ 2.02	+12.32

Table 8: **Evaluation results for Residual RL with the SteerVLA base policy on the Bench2Drive subset.** We evaluate Residual RL on a challenging 16-route subset from Bench2Drive. Results are shown over 3 evaluation seeds.

Route	Base	Residual RL	Δ
signalized-junction-left-turn-001	60.00	60.00	+0.00
enter-actor-flow-004	52.00	86.67	+34.67
non-signalized-junction-right-turn-003	86.67	100.00	+13.33
non-signalized-junction-left-turn-enter-flow-002	48.00	60.00	+12.00
signalized-junction-left-turn-enter-flow-001	60.00	86.67	+26.67
non-signalized-junction-left-turn-001	73.33	46.42	-26.91
signalized-junction-right-turn-004	60.00	60.00	+0.00
pedestrian-crossing-004	80.00	86.67	+6.67
t_-junction-002	80.00	90.00	+10.00
crossing-bicycle-flow-004	40.27	65.95	+25.68
hazard-at-side-lane-002	52.00	100.00	+48.00
merger-into-slow-traffic-v2-005	45.50	45.73	+0.23
vehicle-turning-route-pedestrian-003	78.67	55.52	-23.15
highway-exit-002	75.87	48.54	-27.33
accident-two-ways-002	17.26	29.78	+12.52
interurban-actor-flow-004	96.63	96.94	+0.31
Mean $\pm$ σ	62.89 $\pm$ 4.83	69.93 $\pm$ 6.00	+7.04

Table 9: **Evaluation results for YAY Robot with the SteerVLA base policy on the Bench2Drive subset.** We evaluate YAY Robot on a challenging 16-route subset from Bench2Drive. Results are shown over 3 evaluation seeds.

Route	Base	YAY Robot	Δ
signalized-junction-left-turn-001	60.00	73.33	+13.33
enter-actor-flow-004	52.00	86.67	+34.67
non-signalized-junction-right-turn-003	86.67	67.70	-18.97
non-signalized-junction-left-turn-enter-flow-002	48.00	46.13	-1.87
signalized-junction-left-turn-enter-flow-001	60.00	73.33	+13.33
non-signalized-junction-left-turn-001	73.33	63.49	-9.84
signalized-junction-right-turn-004	60.00	73.33	+13.33
pedestrian-crossing-004	80.00	40.02	-39.98
t_-junction-002	80.00	65.34	-14.66
crossing-bicycle-flow-004	40.27	48.93	+8.66
hazard-at-side-lane-002	52.00	100.00	+48.00
merger-into-slow-traffic-v2-005	45.50	45.50	+0.00
vehicle-turning-route-pedestrian-003	78.67	56.13	-22.54
highway-exit-002	75.87	39.93	-35.94
accident-two-ways-002	17.26	47.62	+30.36
interurban-actor-flow-004	96.63	97.24	+0.61
Mean $\pm$ σ	62.89 $\pm$ 4.83	64.04 $\pm$ 4.86	+1.16

Table 10: **HARP results on the weakest route Bench2Drive subset with $\pi_{0.5}^{AD}$.** We evaluate HARP on a specifically weak subset for $\pi_{0.5}^{AD}$. Results are reported over 3 evaluation seeds.

Route	Base	HARP	Δ
interurban-actor-flow-004	97.24	99.08	+1.84
enter-actor-flow-004	51.33	86.67	+35.34
vehicle-opens-door-two-ways-005	73.33	86.67	+13.34
vanilla-signalized-turn-encounter-green-light-004	59.61	84.08	+24.47
signalized-junction-left-turn-enter-flow-003	73.33	76.67	+3.34
sequential-lane-change-005	86.67	73.33	-13.34
non-signalized-junction-left-turn-enter-flow-002	46.82	67.53	+20.71
signalized-junction-right-turn-004	51.33	67.33	+16.00
vanilla-signalized-turn-encounter-red-light-002	80	66.67	-13.33
pedestrian-crossing-004	66.67	62.36	-4.31
non-signalized-junction-right-turn-001	86.67	60.00	-26.67
signalized-junction-left-turn-001	49.03	56.19	+7.16
crossing-bicycle-flow-004	60.18	56.15	-4.03
merger-into-slow-traffic-v2-001	73.33	44.00	-29.33
non-signalized-junction-left-turn-002	46.82	43.75	-3.07
vehicle-turning-route-pedestrian-005	75	17.64	-57.36
Mean$\pm\sigma$	67.50 $\pm$ 5.11	65.51 $\pm$ 3.26	-1.95

Table 11: **HARP results on the weakest route Fail2Drive subset with $\pi_{0.5}^{AD}$.** We evaluate HARP on a specifically weak subset for $\pi_{0.5}^{AD}$. Results are reported over 3 evaluation seeds.

Route	DS	Seed SD	Δ
generalization-pedestrian-other-blocker-1072	50	100.00	+50.00
generalization-right-construction-1093	10.37	100.00	+89.63
generalization-wall-1097	20.01	100.00	+79.99
generalization-obscured-stop-1048	90	93.33	+3.33
generalization-animals-1083	37.1	88.58	+28.58
generalization-image-on-object-1041	27.85	78.54	+62.01
generalization-fully-blocked-1032	21.04	60.00	+53.92
generalization-bad-parking-1004	30	52.00	+ 22.0
generalization-custom-obstacles-1024	0.29	46.67	+46.30
generalization-construction-pedestrian-1011	27.15	43.33	+13.33
generalization-right-of-way-1056	36	38.49	+2.49
generalization-pedestrians-on-road-1085	5.4	30.83	+25.43
generalization-hard-brake-1036	1.68	25.94	+24.26
generalization-pedestrian-crowd-1069	9.46	14.30	+2.47
generalization-bad-parking-1009	30	9.83	-10.62
generalization-construction-permutations-1019	6.22	3.42	-1.24
Mean	55.33 ± 1.91	+30.17	

Table 12: **Base policies with HARP finetuning on the Bench2Drive routes shared by both sweeps.** Finetuned scores are the mean driving score over eval seeds 1001–1003 (CARLA seed fixed to 0). σ_{seed} is the sample standard deviation of the three per-seed means across the 8 routes; SR is the fraction of route×seed evaluations with DS = 100.

	$\pi_{0.5}^{AD}$			SteerVLA		
Route	Base	+ HARP	Δ	Base	+ HARP	Δ
crossing-bicycle-flow-004	60.18	56.15	−4.03	40.27	100.00	+59.73
enter-actor-flow-004	51.33	86.67	+35.34	52.00	65.33	+13.33
interurban-actor-flow-004	97.24	99.08	+1.84	96.63	97.24	+0.61
non-signalized-junction-left-turn-002	49.45	43.75	−5.70	60.00	44.38	−15.62
non-signalized-junction-left-turn-enter-flow-002	46.82	67.53	+20.71	48.00	71.08	+23.08
pedestrian-crossing-004	66.67	41.76	−24.91	80.00	43.35	−36.65
signalized-junction-left-turn-001	49.03	56.19	+7.16	60.00	71.71	+11.71
signalized-junction-right-turn-004	51.33	67.33	+16.00	60.00	100.00	+40.00
Mean DS ↑	59.01	**64.81**	+5.80	62.11	**74.14**	+12.03
σ_{seed}	8.11	6.87		1.77	5.80	

E HYPERPARAMETERS

E.1 HARP

Table 13: **HARP hyperparameters.** Hyperparameters for fine-tuning the high-level policy with hindsight reflection. Batch weights w are those of the training mixture in Section 3.

Hyperparameter	Value
Policy	
Action chunk length	10 steps
Low-level query interval	3 env steps
High-level CoT query interval	5 env steps
High-level CoT temperature	0.1
Batch weights	
w_{pretrain}	0.5
$w_{\text{corrected}}$	0.45 (2/3 precursor, 1/3 bad)
w_{correct}	0.05
Optimization	
LoRA trainable parameters	326.1M
Learning rate	10^{-5}
KL weight β	0.05
Batch size	64
Gradient steps per update	10
Update interval	200 env steps

E.2 FLUTE

Table 14: **FLUTE hyperparameters.** Coefficients λ_j and safety thresholds ζ_j used for best-of-N selection on each benchmark.

Hyperparameter	Bench2Drive	Fail2Drive
Number of samples N	4	4
High-level policy temperature	1.0	1.0
Low-level policy temperature	0.0	0.0
λ_{crash}	1.0	8.0
λ_{offroad}	0.5	0.5
λ_{traffic}	0.5	0.5
λ_{goal}	0.5	0.5
$\lambda_{\text{progress}}$	0.1	0.1
λ_{correct}	0.5	0.5
ζ_{crash}	0.8	0.8
ζ_{offroad}	0.8	0.8
ζ_{traffic}	0.8	0.8

E.3 BASELINES

Table 15: **Residual RL hyperparameters.** Hyperparameters for the residual SAC baseline (Appendix C.1).

Hyperparameter	Value
Residual action	
Final residual scales $\alpha^{\text{acc}}, \alpha^{\text{steer}}$	0.1, 0.1
Warmup steps	1,000 env steps
Residual ramp length	1,500 env steps
Residual penalty β_{BC}	1.0
SAC	
MLP Size	(256, 256)
Learning rate	3×10^{-4}
Discount γ	0.99
Target-update coefficient τ	0.005
Target-entropy multiplier	0.5
Replay capacity	100,000
Batch size	256
Gradient updates per env step	10
Training env steps	10,000

Table 16: **YAY Robot hyperparameters.** Hyperparameters for the YAY Robot baseline (Appendix C.2).

Hyperparameter	Value
Batch weights	
Offline / online fraction	50% / 50%
Corrective fraction of online data	90%
Precursor fraction of corrective data	2/3
Optimization	
Optimizer	AdamW
Learning rate	10^{-5}
Weight decay	0.1
Gradient clipping	1.0
Batch size / microbatch size	64 / 8
Gradient steps per update	10
Update interval	200 env steps
Training env steps	10,000

F HINDSIGHT REFLECTION PROMPTS

We provide the prompts used with Gemini-3.7-Flash to achieve our results with HARP. These prompts are used for each window.

Listing 11: Task overview block.

```
PRIMARY GOAL of this episode (route `{metadata.get('route', '')}`): [goal
    ]. This is what the ego is TRYING TO ACHIEVE. The routing commands
    below are the means to it, not the end. Judge the episode first and
    foremost on whether it is making real progress toward this goal; an
    episode that follows each command literally but never accomplishes
    the goal has FAILED, and the failure to accomplish it is itself the
    most important BAD event to report.

Overall task - the full sequence of routing commands the ego is given
    over this route, in order (the video may cover only part of it):[plan
     lines]

Where the ego is now: ~{float(here):.0f} m along the route
The routing command in force at the END of this window is: "{current}"
Routing commands issued during this window (the command changes as the
    ego advances; judge each moment against the command in force AT that
    moment)
```

Listing 12: Progress block.

```

Route progress over this window (route completion is a primary objective)
    :

- start: {float(start):.1f}% complete
- end: {float(end):.1f}% complete
- advanced:  {float(delta):.1f}% of the route during this window
- mean ego speed over the last few steps: {float(mean_end_speed):.2f} m/s

- NOTE: the route is NOT complete. If the ego is stopped or barely moving
     with a clear gap ahead (no leading vehicle within stopping distance,
     green/no light, no pedestrian or yield obligation), treat that as
    BAD - it should accelerate and make forward progress. Only reward
    stopping when there is a real reason (red light, stop sign, leading
    vehicle close ahead, pedestrian/cyclist crossing, or a yield).
```

Listing 13: Progress block.

```

Per-timestamp trajectory data (keys are video seconds; each value is the
    vehicle state/controls plus the executed subtask, chain-of-thought
    reasoning, and the prompt the policy received at that moment). ``
    reward_total`` and ``route_progress_pct`` are "given on EVERY
    timestamp; ``ego_speed_mps`` and the ``control_*`` channels are
    sampled every {CONTROL_SAMPLE_EVERY} timestamps to keep this readable
     - where they are absent they simply were not sampled, so read them
    off the neighbouring timestamps or the attached plot rather than
    assuming a change. All numbers are rounded to {TELEMETRY_DECIMALS}
    decimal places. ``reward_total`` is the environment reward earned at
    that step - the objective the policy is being trained on. It is
    dominated by route progress and is reduced by collisions, route/
    traffic infractions, and harsh steering or braking; a sustained near-
    zero or negative stretch marks behavior the environment itself scores
     as bad. Use it as corroborating evidence, not as the verdict: it
    cannot see everything you can in the video, so flag behavior that
    looks wrong even where the reward does not, and say so when the two
    disagree.
```

```
WARNING - ``subtask``, ``reasoning`` and ``prompt`` are the POLICY'S OWN
    output at that moment, not ground truth. They are frequently WRONG,
    and correcting them is the entire point of this review. In particular
     the policy routinely asserts a RED light when the light is green.
    Treat anything it says about a traffic light, a stop sign, a hazard
    or a leading vehicle as an unverified claim, NEVER as evidence, and
    never repeat it back as your own finding. Establish the signal state
    yourself from traffic flow. Where the policy's reasoning contradicts
    what the flow shows, the policy is wrong: say so explicitly and raise
     a BAD event for it:
        f"```json\n{json.dumps(per_timestamp, indent=2)}
```

Listing 14: Plots block.

```
A plot is attached with this video: env reward per step (top) and route
    progress (bottom), both against the SAME video-time axis as the video
     and your event timestamps. Light vertical rules are action-chunk
    boundaries; red rules are collisions. Use it to locate places where
    progress stops climbing, then look at those moments in the video to
    say what caused them
```

Listing 15: Coach prompt.

```
You are reviewing a driving rollout video for an autonomous vehicle
    policy. You are seeing the view of the ego vehicle from the
    perspective of the ego vehicle's camera.
{task_overview_block}
Episode summary:
```json
{metadata_block}
```
{collision_section}{progress_block}{memory_block}{plots_section}{
    per_timestamp_block}
Carefully watch the full video and identify moments where driving
    behavior was clearly
good or clearly bad (lane keeping, speed, turns, collisions/near-misses,
stopping, yielding, route progress, etc.). The collision log above shows
    ground-truth
sensor data - use it to anchor your feedback to the correct timestamps.
    IMPORTANT: the
subtask should always align with the route command plan. For example, if
    the route command plan
if to turn left at the intersection, "turn right" would be a blatant
    violation of the route command plan.
If this is violated, make sure it is in the description of the event.

Everything that is violated in the below steps should be in the
    description of the event.
```

Listing 16: Coach prompt: step 1.

```
** Step 1 **
Determine the state of the vehicle and of the other agents (vehicles,
    pedestrians, cyclists, etc.) in the video.
Ask these questions to guide your analysis:
- What lane in the vehicle in?
- What other vehicles are there? What lanes are they in?
- Is there a leading vehicle? If so, how far ahead is it? Is it stopped
    or moving?
- Are there any pedestrians or cyclists in the video? If so, what are
    they doing?
```

```
- What is the state of the traffic light in front of the vehicle? Is
    there a stop sign?
- IMPORTANT: Always also infer the traffic light states from the traffic
    flow. Vision can be unreliable, so you must use the traffic flow to
    infer the traffic light states.
  For example:
    * cross-traffic moving through the junction => our direction is
        almost certainly red
    * the queue ahead of us discharging, or the lead vehicle pulling away
        => green
    * the lead vehicle stopped at the line with no obstruction ahead of
        it => red
    * pedestrians crossing our path => our direction is red
  Traffic flow is observable from the same camera the car has, so an
      inference of this
  kind is legitimate; a colour you cannot actually see is not. If neither
      the lamp nor
  the traffic flow settles it, say the signal state is UNKNOWN and do not
      raise a
  GOOD/BAD event that depends on it.
- Where is the egovehicle in relation to other vehicles, pedestrians,
    lanes, crosswalks etc.?
- According to the overall task (the ordered routing-command plan above)
    and the current
  routing command in the per-timestamp data, what maneuver should the
      vehicle be
  performing right now (following the road, turning left/right at the
      intersection,
  changing lanes, etc.), and is it in the correct lane and position to do
      so?
```

Listing 17: Coach prompt: step 2.

```
** Step 2 **
Analyze the vehicle's behavior in the video. For example, you can ask
    these questions to guide your analysis:

BEFORE answering ANY question below that mentions a red light, a green
    light, a stop
light or a signal: use the signal state you established in Step 1 FROM
    THE TRAFFIC FLOW.
Do not re-decide the light state by looking at the lamp again - you
    already determined
it, and the lamp is frequently only a few pixels across and washed out.
    Every judgement
below that depends on a signal MUST state, in its description, which flow
     evidence you
relied on (e.g. "cross-traffic was crossing, so our light was red"). If
    Step 1 concluded
the state was UNKNOWN, do not raise a GOOD/BAD event that depends on it
    at all - skip
that question rather than guessing.
PRIORITY - READ BEFORE USING THE CHECKLIST. **Completing the route is the
     primary
objective.** The rules below are CONSTRAINTS on how the vehicle makes
    progress, not goals
in their own right. A vehicle that breaks no rules but does not advance
    along the route has
FAILED, and must be judged as failing. When you are weighing an event,
    ask "did this
advance the route?" first, and "was it safe and legal?" second - both
    must hold, but a
stationary vehicle scores no points for safety.
Excessive caution is a REAL failure mode and you must flag it as BAD,
    just as you would a
```

```
collision. In particular, treat all of the following as BAD whenever
    there is no red
light (as established from traffic flow in Step 1), no stop sign, no
    close leading vehicle,
and no pedestrian/cyclist/yield obligation actually requiring it:
  * stopping or crawling with a clear path ahead;
  * waiting at a junction through a gap that was plainly large enough to
      take - an
    unprotected left or right turn is meant to be COMPLETED, not waited
        out; if the
    vehicle has been stopped for several seconds while cross-traffic has
        a usable gap,
    that is BAD;
  * abandoning a turn part-way, or stopping mid-junction, instead of
      clearing it;
  * braking for an object that is not on the vehicle's path, or that it
      has already passed;
  * creeping well below the speed limit on an open road.
Do NOT invent a hazard to justify a stop. If you cannot point to a
    specific, visible
reason the vehicle had to stop, then the stop was unjustified and the
    event is BAD.
Conversely, do not reward hesitancy: a "safe" event that consists of the
    vehicle sitting
still with the way clear is not GOOD. Only mark GOOD for stopping when
    you can name the
specific hazard or signal that required it.
- Does the vehicle's behavior conflict with the route command plan? (if
    yes, BAD; if no, GOOD).
- Is the vehicle maintaining a safe distance from the front car? (if yes,
     GOOD; if no, BAD)
- Is the vehicle maintaining a safe speed? (if yes, GOOD; if no, BAD)
- Does the vehicle crash with any other vehicles or structures? This is
    directly from the collision log above. (if yes, BAD; if no, GOOD)
- Did the vehicle get dangerously close to any other vehicles and/or
    collide with any other vehicles? (if yes, BAD; if no, GOOD)
- Does the vehicle leave unnecessary gaps between itself and the front
    car? For example, at a
  stop light (identified via the traffic flow in Step 1), the car should
      move forward to be
  closer to the front car to not leave an unnecessary gap (if yes, BAD;
      if no, GOOD)
- Does the vehicle properly wait for a gap in the traffic before changing
     lanes? (if yes, GOOD; if no, BAD)
- Does the vehicle properly yield to other vehicles when necessary? (if
    yes, GOOD; if no, BAD)
- Does the vehicle properly stop at red lights and stop signs? Judge the
    light state from
  the traffic flow you analysed in Step 1, not from the lamp. (if yes,
      GOOD; if no, BAD)
- Does the vehicle properly follow the route and traffic laws? (if yes,
    GOOD; if no, BAD)
- Does the vehicle yield to pedestrians and cyclists when necessary? (if
    yes, GOOD; if no, BAD)
- Does the vehicle follow the rules of the road (turning from left or
    right most lane when making a turn, stopping at stop signs and red
    lights, etc.)? (if yes, GOOD; if no, BAD)
- Is the vehicle making progress along the route toward completion? THIS
    IS THE MOST
  IMPORTANT QUESTION IN THIS LIST. If route progress is below 100% and
      the vehicle is
  stopped or crawling with a clear gap ahead and NO valid reason to stop
      (no red light/stop
  sign AS INFERRED FROM THE TRAFFIC FLOW in Step 1, no close leading
      vehicle, no
```

```
  pedestrian/cyclist, no yield), that is BAD - it should accelerate and
      move forward.
  Check the route-progress plot: any stretch where progress stops
      climbing while the
  vehicle was free to move is a BAD event, and you should say what it
      should have done
  instead (take the gap, complete the turn, resume speed). Conversely,
      resuming motion and
  advancing the route when the way is clear is GOOD. (Do NOT penalize
      stopping that is
  genuinely justified by a red light, stop sign, close leading vehicle,
      or a
  pedestrian/yield - but require that justification to be visible, not
      assumed.)
- FIRST AND MOST IMPORTANT: is the vehicle actually accomplishing the
    PRIMARY GOAL
  stated at the top of this prompt? That goal is inferred from the route'
      s scenario type
  and its routing-command plan together, and it is the point of the
      episode. If the ego
  is drifting away from it - missing the exit it was supposed to take,
      failing to get
  around the obstacle it was supposed to pass, never completing the merge
       - that is the
  single most important BAD event in the window, and the correction must
      say what it
  needed to do to achieve the goal. Obeying the individual routing
      commands while
  failing the goal is NOT success.
- Is the vehicle completing the maneuver the routing command asks for? A
    turn that is
  begun and then abandoned, or a junction the vehicle enters and then
      stalls in, is BAD:
  the correction is to carry the turn through and clear the junction.
- Is the vehicle an appropriate distance from the crosswalk at a red
    light (red as
  established from the traffic flow in Step 1)? If not, it should move
      forward to be
  closer to the crosswalk to not leave an unnecessary gap.
- If the vehicle encounters an obstacle blocking the entire route, does
    it stop entirely before the obstruction? (if yes, GOOD; if no, BAD)
- If the vehicle encounters an obstacle blocking part of the route (one
    lane), does it stop and wait for a gap to go around the obstruction?
    (if yes, GOOD; if no, BAD)
- When the vehicle is waiting to merge into or cross a flow of traffic,
    does it pull out
  as soon as a gap opens? Other traffic in this simulator does NOT yield:
       no oncoming or
  cross-traffic vehicle will slow down or stop to let our vehicle in, and
       no gap will be
  created for it. The only safe gap is the one immediately behind a
      vehicle that has just
  passed, so the correct behavior is to go the moment that vehicle clears
       - hesitating
  there means the next vehicle arrives and the opportunity is gone, which
       is BAD and
  eventually causes a collision when the vehicle finally commits too late
      . Waiting for a
  driver to wave it through or expecting traffic to make room is never
      correct. Judge this
  on the gap that was actually available: pulling out in front of a
      closing vehicle is
  still BAD, but sitting still through a clear gap behind one that has
      just passed is BAD
```

```
  as well, and the correction is to accelerate promptly as that vehicle
      clears.

Return ONLY valid JSON with this schema (no markdown fences):
{{
  "events": [
    {{
      "timestamp_sec": 12.5,
      "label": "GOOD",
      "description": "Explanation of what went well. If this event
          depends on a traffic light or stop sign, it MUST name the
          traffic-flow evidence you used to establish that state. Example
          : 'Held position at the junction; cross-traffic was moving
          through it from t=10.0-14.0s, so our signal was red.'"
    }},
    {{
      "timestamp_sec": 18.0,
      "label": "BAD",
      "description": "Explanation of what went wrong, with the same
          traffic-flow evidence requirement. Example: 'Stayed stopped
          although the lead vehicle pulled away at t=16.5s and the queue
          ahead discharged, so our signal had turned green.'",
      "correction": "Concrete instruction for how to fix the behavior."
    }}
  ],
  "route_divergence": {{
    "diverged": false,
    "timestamp_sec": null,
    "reason": ""
  }}
}}

Rules:
- Use seconds from the start of the video for ``timestamp_sec``.
- ``label`` must be exactly ``GOOD`` or ``BAD``.
- Include ``correction`` only for ``BAD`` events (empty string otherwise)
    .
- Prefer 3-12 high-signal events rather than narrating every second.
- If route progress is below 100% and there is ANY stretch of the window
    where progress
  stopped climbing, you MUST emit at least one event about it - BAD with
      a corrective
  instruction if the vehicle was free to move, GOOD naming the specific
      hazard or signal
  if the halt was genuinely required. Silence about a stall is not an
      option.
- MANDATORY: check the executed SUBTASK against the vehicle's own
    REASONING and against
  the ROUTING COMMAND in force at that moment, and report any
      disagreement between the
  three. All three are in the per-timestamp trajectory data (``subtask``,
       ``reasoning``,
  and ``prompt``, which carries the routing command) and in the routing-
      command plan at
  the top. They are supposed to describe the SAME intent; when they do
      not, the policy is
  about to act on a subtask that contradicts what it reasoned or what the
       route asked, and
  that is a defect to report even if the vehicle happened to drive
      acceptably.
  The authority order is: ROUTING COMMAND first (it is the route's
      instruction and is
  never wrong), then REASONING, then SUBTASK. So:
    * subtask disagrees with reasoning AND routing command -> the SUBTASK
        is wrong. Emit a
```

```
      BAD event whose correction is the subtask that matches them.
          Example: the subtask
      says "turn right", the reasoning says "turn left" and the routing
          command says
      "turn left" -> the subtask must be corrected to turning left.
    * subtask and reasoning agree with each other but disagree with the
        routing command ->
      BOTH are wrong; the correction is the subtask that follows the
          routing command.
    * subtask matches the routing command but the reasoning contradicts
        it -> report it as
      BAD naming the inconsistent reasoning; the correction keeps the
          routing command's
      maneuver.
  Judge against the command in force AT THAT TIMESTAMP, not the one at
      the end of the
  window. Ignore pure wording differences -- "go left at the next
      intersection" and "turn
  left at the junction" are the same intent; only a genuine conflict of
      maneuver,
  direction, or target counts.
- MANDATORY: any event whose description mentions a red/green light, a
    stop light, a
  signal or a stop sign MUST also state the traffic-flow evidence that
      established that
  state (cross-traffic moving, the queue discharging, the lead vehicle
      pulling away or
  holding, pedestrians crossing). "The vehicle stopped at the red light"
      is NOT
  acceptable on its own - the lamp is often a few washed-out pixels and
      cannot be read
  reliably. If you could not establish the state from traffic flow, do
      not emit the
  event at all.

``route_divergence`` - did the ego LEAVE THE PLANNED ROUTE during this
    window?

Set ``diverged: true`` only when the ego committed to a maneuver that
    CONTRADICTS the
routing command in force at that moment - it is now driving somewhere the
     plan never
asked it to go. Concretely: the command was "go left at the next
    intersection" and the
ego went straight through or turned right; the command was to turn and
    the ego drove past
the junction entirely; the ego took a different branch, exit or side road
     than the one the
command named. Use the routing commands listed above (with the metre
    marks and the times
at which each took effect) to decide which command was in force - judge
    the maneuver
against the command active AT THAT MOMENT, not the one at the end of the
    window.

``timestamp_sec`` is the moment the ego became committed to the wrong
    path - the point in
the video where the wrong branch was taken and recovery within the plan
    was no longer
happening (e.g. when it entered the wrong lane of the junction, or when
    its rear axle
cleared the junction on the wrong heading). Give the EARLIEST such moment
    , not when the
consequence became obvious.
```

```
This flag is destructive: everything from ``timestamp_sec`` onward is discarded and
never used to train the policy, because once the ego is off-route the situations it
encounters no longer relate to the task it was given. So be conservative and apply it
ONLY to a genuine wrong-branch maneuver:
- Being slow, hesitant, stopped, or failing to make progress is NOT divergence. It is
  still on the planned route. Emit a BAD event for it instead.
- Lane drift, a wide turn, brief wander over a lane line, clipping a curb, or being
  poorly positioned within the correct road is NOT divergence.
- A collision is NOT by itself divergence (it is handled separately). Only flag it if
  the ego also ended up on a different road than the plan called for.
- If you are NOT confident the ego took a different path than the command asked for,
  set ``diverged: false``. False is the safe answer; when in doubt, choose it.

When ``diverged`` is false, use ``timestamp_sec: null`` and ``reason: ""``. When it is
true, ``reason`` must name the command that was in force and the maneuver actually
performed - e.g. "command from 79 m was 'go left at the next intersection'; the ego drove
straight through the junction at t=6.2s and continued north on the through road."
```

Listing 18: Relabeling prompt.

```
You previously reviewed a short driving rollout window and flagged GOOD and BAD
moments (below). Now assign that credit to the specific action chunks the policy
executed, and for each chunk propose subtasks that would improve the behavior.

PRIORITY. **Completing the route is the primary objective.** Safety and traffic rules are
constraints on how the vehicle makes progress, not goals in their own right - a chunk in
which the vehicle sits still, having broken no rule but advanced the route not at all, is a
FAILING chunk, not a neutral one. Over-conservatism is as real a defect as recklessness and
must be labeled BAD with the same willingness: stopping or crawling with a clear path,
waiting out a gap that was plainly takeable, hesitating at or stalling inside a junction
instead of completing the turn, braking for something not on the vehicle's path, or
creeping far below the speed limit on an open road. Whenever you relabel such a chunk, the
corrected subtask must be one that MOVES the vehicle along the route (take the gap,
complete the turn, resume speed) - never a further-slowing subtask. Do not relabel a chunk
toward stopping unless you can name the specific visible hazard or signal that required it.

Credit is CAUSAL, not just temporal. A bad driving outcome is usually already decided
```

```
several chunks before it becomes visible: the vehicle carried too much speed into the
approach, drifted toward the wrong lane, started braking too late, or committed to a gap
that was never there. Those earlier chunks are ALSO at fault, and fixing them is the only
way to prevent the failure - by the time the BAD event is on screen it is often too late.
So for each BAD event, walk BACKWARD through the preceding chunks and blame the ones that
set it up, giving each a subtask that pre-empts the failure.

Each action chunk spans a fixed number of env steps; the chunk timing (in the same
video seconds as the events) is given below.

Window summary (``route_progress_end_pct`` < 100 means the route is unfinished - making
forward progress toward completion is a primary objective; ``window_reward_total`` is the
environment reward summed over the window):
```json
{json.dumps({k: metadata.get(k) for k in ("episode", "route", "episode_steps", "window_index", "success", "route_progress_start_pct", "route_progress_end_pct", "route_completed", "mean_end_speed_mps", "window_reward_total", "window_reward_mean")}, indent=2)}
```

GOOD/BAD events from the window review (timestamps are video seconds):
```json
{json.dumps(events_payload, indent=2)}
```

Action chunks in this window. ``original_subtask`` is the subtask the policy ACTUALLY
executed over that chunk - it is what you are deciding whether to keep or replace, and the
sequence of them across chunks is what you are smoothing. It is the policy's OWN output
and is frequently WRONG - most commonly it claims a red traffic light when the light was
green, or names a hazard that is not present. Never treat it as evidence about the scene:
the events and established scene above are what settle that. Where an ``original_subtask``
contradicts them, that chunk is BAD and its subtask MUST be replaced rather than kept, and
the replacement must describe what the scene actually showed (for a green light, a subtask
about proceeding or accelerating - not a stopped-at-red one). ``reward_total_sum`` /
``reward_total_mean`` /
``reward_total_min`` are the environment reward earned over each chunk's steps - the
objective the policy is trained on. It rises with route progress and falls with collisions,
route/traffic infractions, and harsh steering or braking. Treat a chunk with a low or
negative reward as corroborating evidence for BAD, and a precursor chunk that still looks
well-paid as a reason to check whether the blame really belongs there. Reward is NOT the
verdict on its own: it is per-step and myopic, so a chunk can be well-paid and still be the
```

```
precursor that made a later failure unavoidable (e.g. carrying speed
    toward an occlusion
earns progress reward right up to the near-miss). The event list remains
    the primary
evidence:
```json
{json.dumps(chunks_payload, indent=2)}
```

{memory_block}
Example subtask phrasings (open vocabulary - reuse verbatim OR write new
    phrases in
the SAME concise style; describe what the vehicle should do, not meta
    commentary). HOWEVER
the new subtasks should NOT reference objects in the scene and should
    focus on the driving behavior of
the vehicle to prevent out of distribution objects from destroying the
    training signal.:
{seed_subtask_block}

REQUIRED REASONING FORMAT. Each `suggested_reasoning` MUST follow the
    template below -
this is a hard constraint, not a stylistic preference. Reasonings that
    deviate are
discarded:

  "Follow the route. <ACTION> <because|to> <justification referencing the
      scene>."

Rules:
  1. Begin with the literal prefix "Follow the route." - always, with no
      preamble.
  2. Follow it with ONE imperative action clause, in the same vocabulary
      as the
     examples: Accelerate / Maintain the speed / Maintain speed /
         Decelerate /
     Remain stopped / Slow down / Stop.
  3. Then justify it with "because ..." or "to ...", naming the relevant
      object in the
     scene (colour + type + position, e.g. "the red wooden fence on the
         left side",
     "the maroon car that is to the front at 8.0 meters") when one is
         relevant.
  4. NEVER write in the first person. No "I", "my", "I must", "I should".
       The reasoning
     is a statement about what the vehicle does, not the driver's inner
         monologue.
  5. NEVER lead with the explanation. Write
     "Follow the route. Remain stopped to prevent collision with the red
         wooden fence on
     the left side." - NOT "I must remain stopped to prevent colliding
         with the red
     wooden fence." and NOT "The fence is blocking the path, so the
         vehicle stops."
  6. One sentence after the prefix. No "Even though ...", no contrastive
      framing.

Example reasoning phrasings (reuse verbatim where one fits, otherwise
    write a new phrase
that obeys the template above):
{seed_reasoning_block}

For EVERY chunk_index above, return:
- label: "GOOD", "BAD", or null.
  * BAD (direct) - a BAD event overlaps this chunk's time range.
```

```
  * BAD (precursor) - no BAD event overlaps this chunk, but it is part of
      the lead-up
    that made a later BAD event unavoidable or hard to avoid. Test each
       preceding chunk
    with: "if the policy had executed a different subtask HERE, would the
        later event
    have been prevented or clearly reduced?" If yes, this chunk is BAD/
       precursor.
  * GOOD - a GOOD event overlaps the chunk AND it is not a precursor to
     any BAD event.
    A chunk that looks smooth in isolation is NOT good if it set up a
       later failure;
    prefer BAD/precursor in that case.
  * null - no event applies and the chunk is not a precursor.
- credit_source: "direct" or "precursor" for BAD chunks (per above); ""
    otherwise.
- rationale: one short sentence tying the chunk to its event. For
    precursor chunks, say
  which later event it leads to and what about this chunk causes it (e.g.
      "Still at
  11 m/s approaching the occluded crosswalk that produces the 8.0s near-
     miss.").
- suggested_subtasks: up to {num_suggestions} subtask phrases that would
    improve or
  reinforce this chunk. For BAD/direct chunks, suggest corrective
     subtasks. For
  BAD/precursor chunks, suggest the subtask that should have been
     executed AT THIS
  MOMENT to pre-empt the later event - the earlier, gentler action (start
      scrubbing
  speed, hold the lane, wait for a real gap), NOT the emergency reaction
     that the
  direct chunk needs. The correction must be justifiable from what is
     observable in
  THIS chunk; do not suggest a subtask that only makes sense with
     hindsight of the
  event. For GOOD chunks, keep the original subtask. If a chunk is BAD
     because the
  vehicle stopped or crawled prematurely while the route is unfinished
     and the way
  ahead is clear (no red light, stop sign, close leading vehicle, or
     pedestrian/yield),
  suggest a subtask that has it accelerate and make forward progress
     along the route.
  The same applies to a chunk that waits out a usable gap at a junction,
     or that
  hesitates or stalls part-way through a turn: the corrective subtask is
     to take the gap
  and carry the turn through, NOT to keep waiting. Prefer the progress-
     making correction
  whenever both a progress-making and a further-slowing subtask would be
     defensible.
  For GOOD or null chunks, keep the chunk's ``original_subtask`` verbatim
      .
  DO NOT USE "reverse" or "back up" or "backwards" OR ANYTHING LIKE THIS
     IN THE SUBTASK.
  This will not be understood by the model and will destroy the training
     signal.
- suggested_reasoning: for BAD chunks whose subtask you are changing (
    direct AND
  precursor), write a fresh, concise chain-of-thought (1-3 sentences,
     present tense)
  that a driver would think BEFORE arriving at the corrected subtask -
     describe what is
```

```
  observed in the scene and why the corrected action is right. For
      precursor chunks this
  must read as anticipation from the current scene ("the crosswalk ahead
      is occluded by
  the parked van, so ease off now"), never as knowledge of the future
      event. This becomes
  the reasoning that leads to the first corrected subtask. Leave it as ""
       for GOOD or
  null chunks (no change needed). The suggested reasoning can reference
      objects in the scene, but the subtasks should not.

**IMPORTANT:** Also try to smooth the subtasks relative to the adjacent
    chunks' subtasks
(compare against each chunk's ``original_subtask`` above, and against the
     subtasks you are
assigning to its neighbours). If there are rapid changes of the subtask,
    you can suggest a
subtask that smooths the transition.

Return ONLY valid JSON (no markdown fences):
{{
  "chunk_credits": [
    {{
      "chunk_index": 3,
      "label": "BAD",
      "credit_source": "precursor",
      "rationale": "Holds full speed toward the crosswalk occluded by the
          parked van, leaving no room to stop for the 2.5s near-miss.",
      "suggested_subtasks": [
        "The vehicle cautiously reduces speed, maintaining steady lane
            keeping."
      ],
      "suggested_reasoning": "A parked van hides the crosswalk on the
          right, so the vehicle eases off the throttle now to keep a
          stopping margin if someone steps out."
    }},
    {{
      "chunk_index": 4,
      "label": "BAD",
      "credit_source": "direct",
      "rationale": "Overlaps the 2.5s near-miss with the pedestrian.",
      "suggested_subtasks": [
        "The vehicle smoothly decelerates to a stop, cautiously adjusting
            course to the left."
      ],
      "suggested_reasoning": "A pedestrian is stepping into the lane
          ahead on the left, so the vehicle must brake now and ease left
          to keep clear rather than continue at speed."
    }}
  ]
}}

Rules:
- Include exactly one entry per chunk_index listed above.
- Use null (not "none") for label when no event applies.
- Keep rationale under 50 words.
- Never exceed {num_suggestions} suggested subtasks per chunk.
- suggested_reasoning must be "" unless the chunk is BAD and you changed
    its subtask.
- credit_source must be "" unless the chunk is BAD.
- A chunk over which route progress did not increase, while the vehicle
    was free to move,
  is BAD (direct) - not null. Treat a flat stretch of route progress with
       no visible
```

```
  hazard as an event in its own right, with a subtask that gets the
      vehicle moving again.
- Only walk blame back as far as it is genuinely actionable: stop once
    the hazard was not
  yet observable from the vehicle's viewpoint, or the situation was still
       comfortably
  recoverable by the chunks that follow. Do not blanket every preceding
      chunk in the
  window - an unrelated chunk that merely happens to come earlier is null
      , not precursor.
```

Listing 19: Example subtasks.

```
"The vehicle cautiously follows the route, maintaining steady lane
    keeping behind the front car.",
"The vehicle smoothly adjusts course to the right while cautiously
    maintaining speed behind the police car.",
"The vehicle maintains a steady course, following the black car ahead at
    a normal pace.",
"The vehicle remains stopped normally at the red traffic light.",
"The vehicle accelerates behind the black car at 8.8 meters, following it
     through the green traffic light.",
"The vehicle accelerated normally through the green traffic light.",
"The vehicle accelerates normally through the green traffic light in 4.8
    meters.",
"The vehicle smoothly decelerates to a stop, cautiously adjusting course
    to the left due to a pedestrian.",
"The vehicle maintains a steady speed while following the route, staying
    behind the dark green SUV.",
"The vehicle smoothly follows the route, cautiously reducing speed
    through the junction with steady lane keeping.",
"The vehicle normally follows the route, maintaining reduced speed behind
     the front black car.",
"The vehicle accelerates steadily to follow the front black car,
    maintaining a straight course.",
"The vehicle turns right aggressively while accelerating through the
    clear junction.",
"The vehicle accelerates normally and makes a sharp left adjustment
    following the route.",
"The vehicle cautiously waits for a gap before normally changing lanes.",
"The vehicle turns right smoothly while accelerating to follow the maroon
     car.",
"The vehicle follows the route, accelerating normally and then making a
    sharp right adjustment.",
"The vehicle cautiously adjusts its course to follow the black car,
    braking if necessary for the walker.",
"The vehicle follows the route, maintaining reduced speed cautiously with
     slight adjustments to the left.",
"The vehicle steadily decelerates to a stop while maintaining a straight
    course.",
"The vehicle cautiously decelerates and adjusts course to follow the
    yellow car ahead.",
"The vehicle remains stopped, patiently waiting behind the stationary
    maroon car.",
"The vehicle remained stopped behind the maroon car 2.8 meters ahead,
    maintaining a steady course.",
"The vehicle decelerates smoothly to stay behind the black car ahead,
    driving cautiously.",
"The vehicle accelerates through a wide left turn, cautiously monitoring
    oncoming traffic.",
"Cautiously adjusting course, the vehicle waits for a gap before changing
     lanes to the lane with oncoming traffic.",
"The vehicle remains stopped behind the black car at 8.8 meters, obeying
    the red traffic light.",
```

```
"The vehicle proceeds through the junction normally due to a green
    traffic light.",
"The vehicle accelerates behind the maroon car, normally proceeding on
    the green traffic light.",
"The vehicle accelerates steadily as the queue ahead discharges through
    the green traffic light.",
"The vehicle follows the route and maintains reduced speed with a steady
    course.",
"The vehicle cautiously follows the route and remains stopped behind the
    black car to the front right.",
"The vehicle remains stopped behind the front black car due to a red
    traffic light, normally maintaining its position.",
"The vehicle follows the front black car forward through the green
    traffic light, normally maintaining its position.",
"The vehicle accelerates normally at a green traffic light.",
"The vehicle remained stopped, steadily maintaining its position behind
    the maroon SUV.",
"The vehicle normally follows the route, accelerating and adjusting right
    .",
"The vehicle cautiously waits for a gap in traffic before changing lanes
    .",
"The vehicle remains stopped normally due to a red traffic light.",
"The vehicle remained stopped normally due to the red traffic light.",
"The vehicle follows the route, maintaining reduced speed with a smooth
    rightward course adjustment.",
"The vehicle stayed stopped, waiting behind the navy car.",
"The vehicle follows the dark green car, accelerating and then sharply
    adjusting course to the left.",
"The vehicle stops steadily behind the dark green car.",
"The vehicle accelerates to follow the dark green SUV at 17.4 meters,
    normally maintaining its lane.",
"The vehicle follows the route with a smooth rightward adjustment, then a
     gradual leftward adjustment, normally maintaining its speed before
    decelerating.",
"The vehicle turns right normally, accelerating to follow the maroon car
    14.6 meters ahead.",
"The vehicle remains stopped to stay behind the black car that is to the
    front in 8.8 meters, driving normally.",
"The vehicle accelerates normally to follow the black car in 9.5 meters
    .",
"The vehicle follows the route with steady lane keeping and normal
    acceleration.",
"Following the route, the vehicle smoothly adjusts left while maintaining
     reduced speed behind the yellow car.",
"The vehicle accelerates normally to follow the gray car ahead,
    maintaining a steady course.",
"The vehicle accelerates and makes a wide right adjustment through the
    junction.",
"The vehicle smoothly follows the route, normally decelerating and
    accelerating to match target speed.",
"The vehicle accelerates normally, maintaining a steady course before
    making a slight adjustment to the left.",
"The vehicle accelerates and then adjusts its course smoothly while
    following the route.",
"The vehicle steadily follows the route, maintaining a reduced speed
    behind the front black car.",
"The vehicle decelerates steadily behind the maroon car, maintaining a
    straight course.",
"The vehicle remained stopped behind the front black car at 9.2 meters,
    normally.",
"The vehicle remains stopped, cautiously following the black car ahead.",
"The vehicle remains stopped, then cautiously accelerates slightly
    forward to maintain position behind the maroon car.",
"The vehicle remained stopped behind the maroon car to the front right in
     8.4 meters.",
```

```
"The vehicle steadily accelerates normally and maintains course along the route.",
"The car is in a driving scenario.",
"The vehicle cautiously accelerates to reach the speed limit with steady lane keeping.",
"The vehicle cautiously decelerates and makes a slight rightward adjustment to follow the black car.",
"The vehicle remains stopped behind the maroon car, normally adhering to the red traffic light.",
"The vehicle cautiously stopped, then smoothly accelerated forward, maintaining its course.",
"The vehicle cautiously waits for a gap before changing lanes.",
"The car follows the route and maintains reduced speed, normally staying behind the orange car at 10.3 meters.",
"The vehicle returns to its route with steady lane keeping, normally.",
"The vehicle remains stopped, steadily maintaining its course behind the maroon car.",
"The vehicle decelerates smoothly to follow the front car, then cautiously accelerates normally.",
"The vehicle follows the route, accelerating normally and making a slight right course adjustment.",
"The vehicle accelerates to follow the dark green SUV ahead, maintaining a steady lane keeping.",
"The vehicle normally follows the route, steadily maintaining speed while making slight right adjustments.",
"The vehicle normally accelerates to follow the front car, then abruptly decelerates to a stop.",
"The vehicle remains stopped, following the route cautiously behind the orange car in 3.1 meters.",
"The vehicle smoothly decelerates and then accelerates to follow the navy car ahead.",
"The vehicle follows the black car, accelerating and then braking cautiously for the pedestrian.",
"The vehicle follows the route, maintaining reduced speed cautiously behind the black SUV at 11.7 meters.",
"The vehicle smoothly decelerates to follow the dark green SUV ahead, then normally accelerates forward.",
"The vehicle remained stopped, cautiously preparing to follow the maroon car ahead.",
"The vehicle follows the route, steadily maintaining speed behind the black car 11.0 meters ahead.",
"The vehicle accelerates normally into the oncoming lane, then makes a slight, smooth course adjustment.",
"The vehicle follows the route, maintaining reduced speed with slight course adjustments.",
"The vehicle turns right smoothly, accelerating to stay behind the silver SUV ahead.",
"The vehicle follows the route with steady lane keeping and normally maintained reduced speed.",
"The vehicle remained stopped, waiting behind the navy car.",
"The vehicle accelerates normally while making a slight adjustment to the left.",
"The vehicle cautiously waits for a gap in traffic before changing lanes.",
"The vehicle normally follows the route, maintaining reduced speed behind the navy SUV ahead with slight right adjustments.",
"The vehicle smoothly adjusts its course leftward while steadily maintaining speed behind the maroon car.",
"The vehicle smoothly adjusts course right while normally maintaining speed behind the dark green car at 10.9 meters.",
"The vehicle cautiously decelerates to stay behind the front black car at 23.6 meters.",
"The vehicle remained stopped, following the black car ahead.",
"The vehicle cautiously makes a left turn after stopping at a stop sign.",
```

```
"The vehicle cautiously decelerates to follow the route, making slight
    adjustments.",
"The vehicle remains stopped normally due to the red traffic light.",
"The vehicle remains stopped normally at a red traffic light.",
"The vehicle normally makes a wide right adjustment to change lanes.",
"The vehicle normally follows the route, maintaining a reduced speed
    behind the navy car to the front right.",
"The vehicle cautiously waits for a gap in traffic before changing lanes
    .",
"The vehicle normally decelerates to stay behind the front right navy car
    , then accelerates.",
"The vehicle maintains steady lane keeping, cautiously following the
    silver car ahead at 19.2 meters.",
"The vehicle follows the route, accelerating normally to keep pace with
    the dark green SUV ahead.",
"The vehicle remains stopped normally due to the red traffic light in 4.8
     meters.",
"The vehicle accelerates normally at the green traffic light.",
"The vehicle moves off smoothly as the traffic light turns green,
    maintaining steady lane keeping.",
```