R2V AGENT: TEACHING SLMS WHEN TO ASK FOR HELP

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient agentic systems should incur expensive frontier-model costs only on decisions where a cheaper local model is likely to fail. Existing LLM cascades usually route whole queries before execution, but task difficulty shifts mid-trajectory - after flaky tool calls, truncated observations, or compounding local errors - making pre-execution routing brittle. We introduce **R2V-Agent**, a risk-calibrated SLM-LLM routing framework for interactive agents. R2V combines four components: a distilled small language model (SLM) policy, a stronger teacher LLM, a lightweight process verifier that scores candidate actions at each step, and a calibrated step-level router. The router is our central contribution: after the SLM is trained, it estimates residual failure risk at each step and escalates only when teacher intervention is warranted. To make the routing problem well-defined, we first train a stable local SLM using a standard offline pipeline: behavioral cloning (BC) on teacher trajectories, followed by verifier-guided Direct Preference Optimization (DPO) with consistency regularization. The router is then trained on this fixed policy's residual failures using Brier-calibrated probability estimation and a Conditional Value-at-Risk (CVaR)-constrained objective that penalizes worst-case failures across perturbation seeds. Across HumanEval+, TextWorld, and TerminalBench with four SLM backbones, R2V improves the reliability-cost frontier: it achieves 94.3% HumanEval+ success with 0.60% LLM escalation, recovers TextWorld from 64.6% SLM-only success to 98.2% at 41.7% escalation, and reaches 93.3% TerminalBench success at 33.9% LLM calls, roughly half the heuristic-router cost.

1 INTRODUCTION

Autonomous language-model agents increasingly operate in interactive settings such as software engineering (Cognition, 2024; Yang et al., 2024b), web navigation (Zhou et al., 2024a), tool use, and system administration. Reliability in these environments often depends on expensive frontier LLMs, especially when tasks require long-horizon planning, recovery from failed tool calls, or reasoning under incomplete observations (Mialon et al., 2023). At the same time, inference-time scaling and process supervision suggest that additional computation is most valuable when allocated to the right intermediate decisions rather than spent uniformly across all steps (Snell et al., 2025; Xi et al., 2025b). This raises a natural systems question: can a cheap local SLM execute routine agent steps while a stronger LLM is invoked only when the current decision is genuinely high risk?

Existing LLM cascades and routers make progress toward cost reduction, but most are designed for static, query-level allocation. Systems such as FrugalGPT and RouteLLM decide which model should answer a request before execution, or cascade after a full response has already been generated (Chen et al., 2024; Ong et al., 2025; Dekoninck et al., 2025; Ding et al., 2024b). This abstraction is poorly matched to agentic POMDPs, where difficulty is trajectory-dependent. A task may begin in-distribution, then become risky after a flaky tool call, truncated observation, prompt injection, or compounding local error. Routing the whole task to the SLM is brittle, while routing the whole task to the LLM is unnecessarily expensive. Figure 1 illustrates the difference: monolithic frontier execution is robust but costly, query-level routing is cheap but blind to mid-trajectory failures, and R2V-Agent escalates only at risk-critical steps.

We propose **R2V-Agent**, a risk-calibrated framework for step-level SLM-LLM routing in interactive agents. R2V factorizes execution into four components: an efficient SLM policy π_θ, a stronger teacher

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

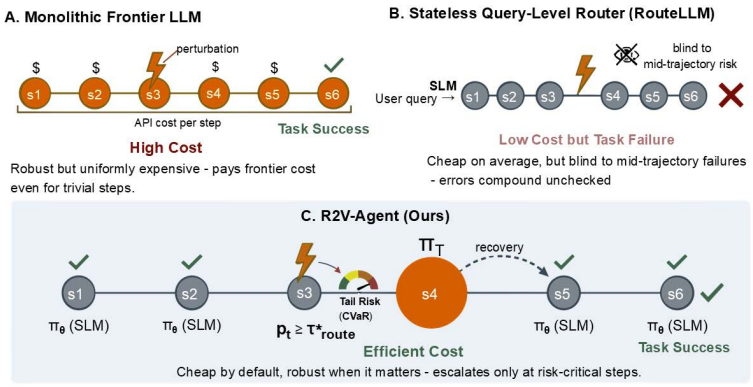


Figure 1: Traditional agentic workflows versus R2V-Agent. Monolithic frontier-LLM execution is robust but expensive; stateless query-level routing is cheaper but blind to mid-trajectory risk; R2V performs step-level escalation only when verifier-grounded risk exceeds the routing threshold.

LLM π_T, a lightweight process verifier $V_\phi(x_t, a_t)$ that scores candidate actions, and a router $r_\psi(f_t)$ that estimates whether the current step should be escalated. The router is the main methodological contribution. It learns when a *fixed deployed SLM* should defer to the teacher based on residual step risk, rather than predicting the difficulty of the initial query.

To make this routing problem well-defined, we first train a stable local SLM with an offline distillation pipeline. The SLM is warm-started with behavioral cloning on successful teacher trajectories, including their perturbed variants. It is then refined with verifier-guided DPO and consistency regularization across perturbation instantiations, while keeping the BC-trained policy frozen as the DPO reference. We use this setup because it is stable and reproducible, not because BC or DPO are themselves new optimization contributions. Only after the distilled SLM policy is fixed do we train the router on that policy's residual failures. This separation ensures that the main experiments isolate compute allocation rather than entangling routing with changes to the local policy.

The router is trained as a calibrated probabilistic decision interface. Brier regularization encourages $r_\psi(f_t)$ to estimate the conditional probability that continuing with the SLM will fail, while a CVaR-constrained objective discourages routers that are cheap on average but brittle under tail perturbation seeds (Chow & Ghavamzadeh, 2014; Tamar et al., 2015). The resulting deployment rule is simple: execute locally by default, but escalate when calibrated residual risk exceeds the routing threshold and budget remains.

Empirically, R2V improves the reliability-cost frontier across HumanEval+, TextWorld, and TerminalBench. Averaged over Gemma-9B, LLaMA-3.1-8B, Qwen2.5-7B, and Qwen2.5-14B backbones, R2V reaches 94.3% success on HumanEval+ with only 0.6% LLM escalation; recovers TextWorld to the 98.2% LLM success ceiling at 41.7% escalation, close to the 35.4% oracle rate; and improves TerminalBench success from 81.1% to 93.3% while using 33.9% LLM calls, compared with 66.0% for the heuristic router at comparable reliability.

Contributions. We formulate SLM–LLM agent routing as a stateful step-level allocation problem in perturbed POMDPs. We introduce a residual-risk routing objective that combines Brier-calibrated probability estimation with CVaR-constrained tail-risk control. We evaluate the router on top of a fixed BC→DPO recovery-distilled SLM across three benchmarks and four backbones, showing that calibrated step routing substantially reduces frontier-model calls while preserving high task success.

2 Related Work

R2V-Agent builds on efficient model routing, process supervision, recovery-oriented policy improvement, and risk-sensitive decision making; Appendix D provides a more extensive review. Existing LLM cascades and routers reduce inference cost by selecting which model should answer a request, either before generation or after a cheaper model has produced a response (Chen et al., 2024; Dekoninck et al., 2025; Ong et al., 2025; Ding et al., 2024a; Zhuang et al., 2024; Hu et al., 2024). These methods

are effective for static request-level allocation, but they treat difficulty as a property of the initial prompt or completed response. R2V targets a different setting: in interactive agents, risk evolves after tool calls, partial observations, prompt perturbations, and earlier local actions. We therefore route at the step level, conditioning escalation on the unfolding agent state and verifier-grounded evidence about the deployed SLM's residual failures.

R2V also relates to process reward models and verifiers for multi-step reasoning and agentic control (Wang et al., 2024; Snell et al., 2025; Xi et al., 2025b). We use the verifier not only to score candidate actions, but also to construct recovery preferences and routing features. This connects to preference optimization and recovery-style training (Rafailov et al., 2023; Kumar et al., 2025); however, R2V uses the standard BC→DPO scaffold only to obtain a stable fixed SLM, and trains the router afterward on that. Finally, R2V draws on calibration and risk-sensitive RL: Brier calibration gives a probabilistic residual-risk interface (Guo et al., 2017), while CVaR discourages routers that are cheap on average but brittle under perturbations (Chow & Ghavamzadeh, 2014; Tamar et al., 2015). Table 14 summarizes the distinction: prior routers are largely query-level and stateless, while R2V combines stateful step routing, process verification, perturbation recovery, and CVaR-calibrated escalation.

3 PRELIMINARIES

We model agent-environment interaction as a Partially Observable Markov Decision Process (POMDP) (Sutton & Barto, 2018) augmented with a latent perturbation seed z:

$$\begin{aligned} \mathcal{M}_z &= (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{T}, \mathcal{O}_z, R, \gamma), \\ \mathcal{O}_z : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \Delta(\mathcal{O}), &\qquad x_t = (G, o_{\le t}, a_{<t}). \end{aligned} \tag{1}$$

Here $z \in \mathcal{Z}$ captures unobserved stochastic factors such as tool flakiness, latency spikes, truncated logs, or prompt corruption; G is the task goal. A frontier policy $\pi(a \mid x)$ is robust but expensive at every step, whereas a local SLM is efficient but brittle under out-of-distribution perturbations. R2V-Agent therefore factorizes execution into an efficient base policy $\pi_\theta(a \mid x)$, a stronger teacher policy $\pi_T(a \mid x)$, a lightweight process verifier $V_\phi(x, a) \in [0, 1]$, and a router $r_\psi(f_t) \in [0, 1]$ that decides whether to execute locally or escalate to the teacher.

R2V is trained as an *ordered pipeline* rather than a joint multi-loss objective. First, an offline trajectory pool is constructed by replaying teacher demonstrations under perturbation operators, yielding a mix of unperturbed and perturbed trajectories. Behavioral cloning (BC) on the episode-success subset of this pool produces a nominal policy π_θ^{BC}. Second, this fixed BC policy generates candidate actions on the same trajectory contexts, which the verifier ranks into preference pairs, after which the SLM is refined with DPO and consistency regularization using the BC policy as the frozen DPO reference. Only after the distilled SLM is fixed do we train the router on its residual high-risk states. This separation makes the router calibrated to the failure modes of the policy actually deployed, and isolates compute allocation from changes in local-policy training.

3.1 VERIFIER-GUIDED DISTILLATION ON PERTURBED TOPOLOGIES

Let $\mathcal{D}_{\mathrm{exp}} = \{(x_t, a_t^\star)\}$ denote expert demonstrations collected from π_T. Before BC training, we replay these trajectories under perturbation operators $\mathcal{P}_k$ to construct a perturbed set $\mathcal{D}_{\mathrm{pert}}$, that mimic tool failures, missing observations, adversarial strings, or other stochastic disruptions and then train BC on the episode-success subset of $\mathcal{D}_{\mathrm{traj}} = \mathcal{D}_{\mathrm{exp}} \cup \mathcal{D}_{\mathrm{pert}}$:

$$\mathcal{L}_{\mathrm{BC}}(\theta) = -\mathbb{E}_{(x_t, a_t^\star) \sim \mathcal{D}_{\mathrm{traj}}} \left[\log \pi_\theta(a_t^\star \mid x_t) \right]. \tag{2}$$

At each x_t' from $\mathcal{D}_{\mathrm{traj}}$, the BC policy proposes

$$a_t^{(1)}, \ldots, a_t^{(K)} \sim \pi_\theta^{\mathrm{BC}}(\cdot \mid x_t'),$$

which are scored by V_ϕ; when needed, we also query π_T for a recovery action a_t^T. We then form preference pairs (a_t^+, a_t^-), where a_t^+ is verifier-approved or teacher-recovered and a_t^- is a lower-quality BC-sampled alternative. Thus, preference learning targets the BC policy's own recoverable failure modes rather than generic preference data. Appendix A gives auxiliary results on verifier-guided candidate selection and bounded pairwise verifier noise.

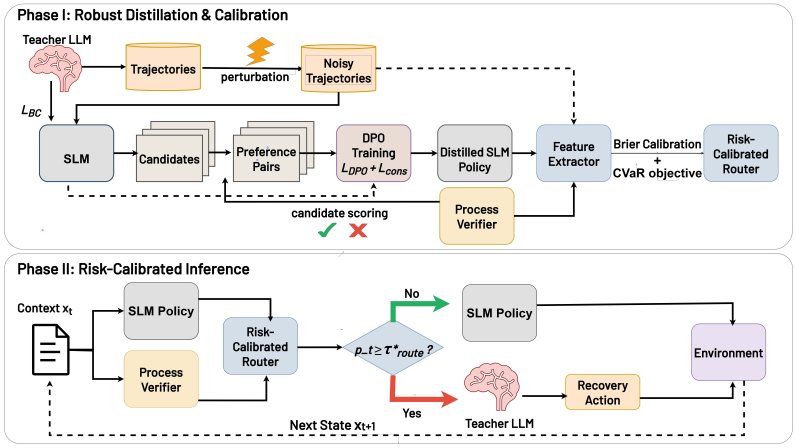


Figure 2: **R2V-Agent pipeline.** *Phase I:* Teacher trajectories are perturbed to train a BC-initialized SLM with verifier-guided DPO and consistency regularization; verifier and policy features then train a Brier-calibrated and CVaR-calibrated router. *Phase II:* At inference, the SLM acts by default, while the teacher LLM is invoked only when the router's residual-risk estimate exceeds τ^*_{route}.

After constructing preference pairs, we freeze the BC policy as the DPO reference, $\pi_{\text{ref}} = \text{stopgrad}(\pi_\theta^{\text{BC}})$, initialize from π_θ^{BC}, and optimize DPO on perturbed states:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x'_t, a^+, a^-)} \left[\log \sigma \left(\beta \left[\log \frac{\pi_\theta(a^+ \mid x'_t)}{\pi_{\text{ref}}(a^+ \mid x'_t)} - \log \frac{\pi_\theta(a^- \mid x'_t)}{\pi_{\text{ref}}(a^- \mid x'_t)} \right] \right) \right] \tag{3}$$

This offline objective shifts the SLM toward perturbation-recovery actions without unstable multi-turn RL. To further stabilize behavior under partial observability, we regularize action distributions across different perturbation realizations of the same latent state:

$$\mathcal{L}_{\text{cons}}(\theta) = \mathbb{E}_{t,z,z'} \left[\text{JSD} \left(\pi_\theta(\cdot \mid x_t^{(z)}) \,\middle\|\, \pi_\theta(\cdot \mid x_t^{(z')}) \right) \right] \tag{4}$$

This encourages perturbation-invariant action distributions (Lin, 1991; Qiang et al., 2024b; Wang et al., 2025b); Appendix A shows that it controls the transfer gap between seen and unseen perturbation seeds.

4 Risk-Calibrated Routing & Verifier-Distillation (R2V)

We first obtain a fixed local policy through behavioral cloning on successful clean and perturbed teacher trajectories, followed by verifier-guided DPO and perturbation-consistency regularization, as defined in Equations equation 2–equation 4. This standard recovery-distillation scaffold is used only to produce a stable local policy. All routing experiments train and evaluate the router after this policy is fixed; the core R2V contribution begins with residual-risk estimation below.

4.1 Risk-Calibrated Trajectory Routing via CVaR

For each context x_t, the fixed SLM proposes K candidate actions, the verifier scores them, and a compact risk feature vector is extracted:

$$f_t = \Phi(x_t, \pi_\theta, V_\phi) \in \mathbb{R}^{15}. \tag{5}$$

The features include token-level entropy and log-probability statistics; verifier-score mean, standard deviation, spread, best score, and worst score over candidates; candidate consistency and semantic entropy; horizon fraction and absolute step index; normalized context length; and a goal-length proxy. Benchmark and perturbation-type one-hot features are implemented but disabled in the reported experiments.

The routing label is derived from the downstream SLM episode outcome, not from counterfactual teacher success. For a logged SLM rollout $\tau^{\text{SLM}}(z)$ under perturbation seed z, let $S_{\text{SLM}}(z) \in \{0, 1\}$

denote binary episode success. Each visited context receives

$$y_t = \mathbf{1}[S_{\mathrm{SLM}}(z) = 0]\,. \tag{6}$$

Thus, y_t is not a ground-truth step-correctness label; it marks contexts on SLM-failure trajectories. The router $r_\psi(f_t)$ estimates the residual probability that continuing with the deployed SLM will fail the episode from the current local evidence.

For smooth offline training, the router probability

$$p_t = r_\psi(f_t) \tag{7}$$

relaxes the hard escalation decision. The cost-sensitive surrogate and Brier calibration loss are $\ell_{\mathrm{route}}(p_t, y_t) = c_{\mathrm{SLM}}(1 - p_t) + c_{\mathrm{LLM}}p_t + \kappa y_t(1 - p_t)$,

$c_{\mathrm{SLM}} < c_{\mathrm{LLM}},\ \kappa > 0,$ To control tail failures across perturbation

$\mathcal{L}_{\mathrm{Brier}}(\psi) = \mathbb{E}_{(f_t,y_t)\sim\mathcal{D}_{\mathrm{route}}}\left[(r_\psi(f_t) - y_t)^2\right].$

seeds, we aggregate stepwise routing losses into a seed-level risk

$$\widetilde{R}_\psi(z) = \frac{1}{|\tau^{\mathrm{SLM}}(z)|} \sum_{t\in\tau^{\mathrm{SLM}}(z)} \ell_{\mathrm{route}}(r_\psi(f_t), y_t)\,, \tag{8}$$

and train the router with the differentiable CVaR-constrained Lagrangian

$$\min_{\psi}\max_{\lambda\geq 0}\ \mathbb{E}_z\left[\widetilde{R}_\psi(z)\right] + \lambda\left(\mathrm{CVaR}_\alpha\left(\widetilde{R}_\psi(z)\right) - \epsilon\right) + \lambda_B\mathcal{L}_{\mathrm{Brier}}(\psi). \tag{9}$$

The hard rule $d_t = \mathbf{1}[r_\psi(f_t) \geq \tau_{\mathrm{route}}]$ is used only for validation-time threshold selection and test-time execution in Algorithm 2; we do not differentiate through the threshold, environment transition, or remaining-budget gate. Optimization is therefore offline and smooth, while evaluation uses the actual thresholded routed policy.

Appendix A shows that Brier minimization recovers the conditional episode-failure risk

$$q^\star(f) = \mathbb{P}(y_t = 1 \mid f_t = f). \tag{10}$$

For the unconstrained one-step surrogate in Eq. equation 4.1, the Bayes threshold before clipping is

$$\bar{\tau}_{\mathrm{route}} = \frac{c_{\mathrm{LLM}} - c_{\mathrm{SLM}}}{\kappa}.$$

Since router probabilities lie in $[0, 1]$, the feasible threshold is

$$\tau^\star_{\mathrm{route}} = \min\left\{1, \max\left\{0, \frac{c_{\mathrm{LLM}} - c_{\mathrm{SLM}}}{\kappa}\right\}\right\}. \tag{11}$$

When $0 \leq c_{\mathrm{LLM}} - c_{\mathrm{SLM}} \leq \kappa$, this reduces to the unclipped threshold. If $c_{\mathrm{LLM}} - c_{\mathrm{SLM}} > \kappa$, escalation is too expensive for any calibrated risk below one; if $c_{\mathrm{LLM}} \leq c_{\mathrm{SLM}}$, the surrogate always prefers escalation. Algorithm 2 additionally enforces a finite LLM budget, so the remaining-budget check acts as a feasibility projection during online execution.

For the plug-in hard router $\widehat{d}_t = \mathbf{1}[r_\psi(f_t) \geq \tau^\star_{\mathrm{route}}]$, the excess one-step routing cost is bounded by calibration error:

$$\mathbb{E}\left[\ell_{\mathrm{route}}(\widehat{d}_t, y_t) - \ell_{\mathrm{route}}(d^\star_t, y_t)\right] \leq \kappa\mathbb{E}\left[|r_\psi(f_t) - q^\star(f_t)|\right] \tag{12}$$

This makes calibrated residual-risk probabilities, rather than ad hoc entropy thresholds, the central routing interface.

4.2 Inference with Risk-Calibrated Escalation

At test time, the distilled SLM executes by default, while the teacher LLM is reserved for steps whose calibrated residual risk exceeds the routing threshold and for which budget remains. Algorithm 2 summarizes the full procedure: the SLM proposes actions, the verifier scores local quality, and the router allocates expensive teacher compute only at predicted high-risk steps.

5 EXPERIMENTS

We evaluate whether R2V-Agent preserves frontier-model reliability while invoking the LLM only on high-risk agent steps. Following the step-level execution interface in Figure 1, Figure 2, and Algorithm 2, the SLM proposes candidate actions, the process verifier scores local action quality, and the router escalates only when the predicted residual risk exceeds the calibrated threshold.

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate R2V on three sequential decision-making benchmarks that stress different forms of agent risk: code generation, text-based embodied control, and terminal engineering. All benchmarks use a task-level $70/15/15$ train/validation/test split with seed 42, and each clean trajectory is replayed under 5 independently sampled perturbation seeds. On **HumanEval+**, we use all 164 Python programming tasks and cast each problem as a multi-step environment with `write_code`, `test`, and `submit` actions; Gemini 3 Flash Preview provides 820 clean teacher trajectories. On **TextWorld**, we use ALFWorld (Shridhar et al., 2021b), a household-control benchmark with 250 tasks across six task families, where agents issue textual commands such as *go to*, *take*, *put*, *clean*, and *heat*; Gemini 3 Flash Preview provides $800/200/250$ train/validation/test episodes. HumanEval+ and TextWorld use four perturbation families: tool flakiness, partial observability, prompt injection, and distractor observations. On **TerminalBench** (Merrill et al., 2026), we use 89 real-world shell tasks from `yoonholee/terminalbench-trajectories` (yoonholee, 2026), including toolchain compilation, git-history repair, gRPC implementation, and numerical-kernel optimization. TerminalBench teacher trajectories are collected from Claude Opus 4.6, GPT-5, and Gemini 3.1 Pro, and its perturbations are restricted to tool flakiness and partial observability because semantic prompt injection and distractors are less meaningful for shell execution.

Verifiers and router. Each benchmark uses a lightweight CPU-only process verifier $V_\phi(x_t, a_t) \in [0, 1]$ that provides local evidence rather than oracle supervision. HumanEval+ combines smoke-test execution with structural code checks; TextWorld scores observation quality, action type, goal alignment, and repetition; and TerminalBench parses terminal output for success/failure signatures, command category, repetition, and completion intent. Router labels are derived from downstream SLM-only episode outcomes: a visited context receives $y_t = 1$ if the fixed SLM rollout containing that context fails the task, and $y_t = 0$ otherwise. Thus, labels mark residual SLM failure risk rather than counterfactual teacher correctness at each step. Full verifier definitions and label statistics are given in Appendix B.

The router $r_\psi : \mathbb{R}^{15} \to [0, 1]$ is a two-layer MLP with hidden widths 128 and 64, GELU activations, dropout $p = 0.2$, Brier calibration, and post-hoc temperature scaling (Guo et al., 2017). Its inputs summarize SLM uncertainty, top-K candidate log-probabilities, verifier-score statistics, candidate diversity, horizon position, and context length. We use $K = 5$ candidates sampled with vLLM (Kwon et al., 2023); unless otherwise stated, R2V uses $\alpha = 0.20$, $\varepsilon = 0.10$, and $c_{\mathrm{LLM}}/c_{\mathrm{SLM}} = 50$.

Baselines. We compare against **SLM-only**, **LLM-only**, **Entropy Router**, **Heuristic Router**, and an offline **Oracle Router**. The entropy router escalates using a validation-calibrated token-entropy threshold, while the heuristic router uses non-learned verifier rules. The oracle is a diagnostic upper bound: it has hindsight access to the SLM-only episode outcome and marks contexts on failed SLM rollouts before execution. It is therefore not an attainable policy under Algorithm 2, since an online router must infer risk from f_t without knowing the future trajectory. Within each benchmark, all deployable methods share the same distilled SLM and the same fixed verifier, and differ only in the routing decision; the verifier differs across benchmarks only because the observation modality differs (code execution, text state, and terminal output). The router consumes only verifier score statistics, not benchmark-specific verifier logic. The main results in Table 1 therefore evaluate step-level compute allocation, not benchmark-specific verifier engineering or the standard BC/DPO distillation scaffold.

5.2 MAIN RESULTS

Table 1 reports success rate (SR) and LLM escalation rate averaged across four SLM backbones: Gemma-9B (Team, 2024), LLaMA-3.1-8B (Grattafiori et al., 2024), Qwen2.5-7B, and Qwen2.5-14B (Qwen et al., 2025). Since all deployable methods share, within each benchmark, the same distilled

Table 1: **Benchmark-level results** averaged over four SLM backbones under noisy evaluation. "LLM%" is the fraction of steps escalated to the teacher. R2V uses α=0.20, ε=0.10. The oracle is a non-deployable hindsight diagnostic.

Method	HumanEval+		TextWorld		TerminalBench	
	SR	LLM%	SR	LLM%	SR	LLM%
SLM-only	91.9	0.0%	64.6	0.0%	81.1	0.0%
Entropy Router	92.9	0.8%	64.6	0.0%	87.4	4.1%
Heuristic Router	97.4	26.6%	98.4	99.9%	95.0	66.0%
R2V (ours)	**94.3**	**0.6%**	**98.2**	**41.7%**	**93.3**	**33.9%**
Oracle Router	98.6	8.1%	98.6	35.4%	97.8	18.4%
LLM-only	98.8	100%	98.7	100%	98.1	100%

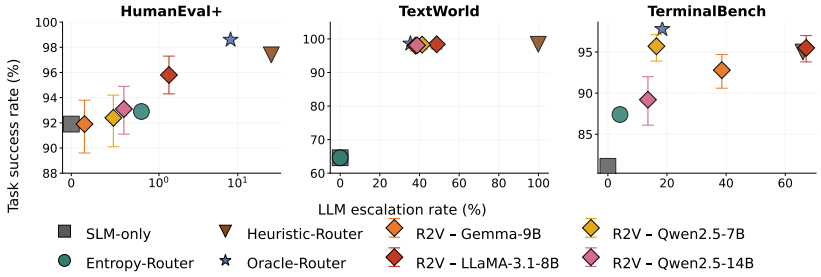


Figure 3: **Cost-performance Pareto frontier.** Each R2V point corresponds to one SLM backbone with 95% bootstrap confidence intervals. R2V gives near-free gains on HumanEval+, closely tracks the oracle on TextWorld, and recovers substantial SR for weaker TerminalBench backbones while remaining below heuristic-router cost.

SLM and the same fixed verifier, these results isolate the routing decision. The theory in Section 4 predicts that a calibrated router should escalate only when the estimated residual failure risk justifies the teacher call, while the CVaR constraint should avoid policies that are cheap on average but brittle under perturbation seeds. The results match this behavior: R2V uses little teacher compute in low-failure regimes, escalates substantially when hidden state risk is high, and remains below heuristic-router escalation on the most heterogeneous benchmark.

HumanEval+. HumanEval+ is a low-failure regime: only 8.11% of steps belong to failed SLM episodes. Consistent with the calibrated-threshold view, R2V rarely escalates. It improves average SR from 91.9% to 94.3% while using only 0.6% LLM calls. The entropy router gives a smaller gain at similar cost, whereas the heuristic router reaches higher SR but requires 26.6% LLM calls. Thus, on self-correcting code tasks, most of the benefit comes from identifying a small set of high-risk steps rather than broadly increasing teacher usage.

TextWorld. TextWorld is the clearest test of state-dependent risk under partial observability. The SLM-only policy reaches only 64.6% SR, and the entropy router remains at 64.6% with 0.0% escalation, showing that token uncertainty alone misses many observation-induced failures. R2V reaches 98.2% SR with 41.7% LLM calls, within 0.4 points of the hindsight oracle's 98.6% SR and within 0.5 points of LLM-only execution. The heuristic router reaches 98.4% SR, but at 99.9% escalation, effectively collapsing to LLM-only execution. This supports the residual-risk formulation: the useful signal is not just local entropy, but calibrated risk from verifier, uncertainty, and trajectory-state features.

TerminalBench. TerminalBench is the most heterogeneous setting. SLM-only performance is much lower at 81.1% SR, and the oracle gap is larger than on the other benchmarks. R2V improves SR to 93.3% while using 33.9% LLM calls, roughly half the heuristic router's 66.0% escalation rate. The remaining gap to the oracle, which reaches 97.8% SR with 18.4% LLM calls, indicates that some shell-task failures are identifiable in hindsight but not yet fully captured by the current online risk features. Per-backbone results in Appendix C.1 show that R2V both recovers weaker models and reduces unnecessary escalation for stronger models.

Table 2: **Weighted execution cost per episode** (lower is better), averaged over four SLM backbones. Cost charges 1 unit per local step and 50 units per escalated teacher step. The last two rows give R2V's cost reduction relative to LLM-only and the heuristic router.

Method	HumanEval+	TextWorld	TerminalBench
SLM-only	10.9	12.4	20.0
Entropy Router	16.4	12.4	71.4
Heuristic Router	162.4	620.9	645.1
R2V (ours)	**17.1**	**435.1**	**486.9**
LLM-only	544.6	621.2	1001.8
R2V vs. LLM-only	-96.9%	-30.0%	-51.4%
R2V vs. Heuristic	-89.5%	-29.9%	-24.5%

Figure 3 summarizes the same trend as a cost-performance frontier. HumanEval+ occupies a low-risk regime where escalation is rarely needed; TextWorld requires roughly 40% escalation because failures are strongly tied to hidden state and observation quality; and TerminalBench exposes residual headroom, with the hindsight oracle reaching 97.8% SR at only 18.4% escalation. This oracle point should be interpreted as a diagnostic estimate of recoverable failure mass in logged SLM rollouts, not as a deployable policy with online access to future outcomes.

5.3 Cost and Tail-Risk Analysis

The escalation rate in Table 1 does not by itself show that R2V is cheaper end-to-end, so we report the weighted execution cost per episode and the worst-seed reliability of each method. Cost charges one unit per local SLM step and $c_{\mathrm{LLM}}/c_{\mathrm{SLM}} = 50$ units per escalated teacher step, matching the router training objective; it does not separately bill the $K{=}5$ candidate sampling or verifier scoring, which run locally on CPU. All numbers are averaged over the four SLM backbones and computed over the five perturbation seeds.

Table 2 shows that R2V's low escalation rate translates into real cost savings. On HumanEval+, R2V costs 17.1 units per episode, a 96.9% reduction relative to LLM-only (544.6) and 89.5% relative to the heuristic router (162.4). On TextWorld, R2V costs 435.1 units, 30.0% below LLM-only and 29.9% below the heuristic router, at the same reliability. On TerminalBench, R2V costs 486.9 units, 51.4% below LLM-only and 24.5% below the heuristic router. Because R2V escalates only 0.6%, 41.7%, and 33.9% of steps, its teacher-token bill is that same fraction of LLM-only's for any cost ratio, so these savings grow as the teacher becomes more expensive relative to the SLM.

Worst-seed robustness. R2V also improves worst-seed SR over SLM-only on all three benchmarks, with worst-seed performance remaining close to its mean SR; full seed-level results are reported in Appendix C.2.

Router gains are not explained by verifier quality. The process verifiers differ sharply in accuracy across benchmarks: episode-level AUROC is 0.856 on HumanEval+, ≈ 0.78 on TextWorld, and only 0.508 on TerminalBench (Appendix B.2, Table 4). Despite the near-chance TerminalBench verifier, R2V still improves success rate from 81.1% to 93.3%. The routing gains therefore come from the learned, calibrated combination of uncertainty, verifier statistics, and trajectory-state features, not from a strong hand-engineered verifier.

5.4 Ablations

Table 3 reports a compact ablation of perturbation consistency and router features; full sweeps are provided in Appendix C.2. Because the underlying SLM is fixed across routing methods, these rows isolate how regularization and inference-time features change the reliability–cost operating point.

Consistency regularization mainly changes the cost–reliability operating point. On HumanEval+, moving from $\lambda_{\mathrm{cons}} = 0$ to 0.20 lowers escalation from 1.1% to 0.4% but also reduces SR from 94.0% to 92.4%. TextWorld is stable, improving slightly from 97.8% to 98.3% SR. TerminalBench benefits most: $\lambda_{\mathrm{cons}} = 0.20$ increases SR from 93.8% to 95.7% while reducing LLM calls from 35.7% to 16.3%. This is consistent with the perturbation-invariance motivation of the JSD term: the largest gains appear in the setting with high observation and tool-output variability.

Table 3: **Compact ablation summary.** Cells report SR / LLM%. Consistency and feature ablations are evaluated on Qwen2.5-7B; feature masking is applied at inference time without retraining.

Ablation	Variant	HumanEval+	TextWorld	TerminalBench
Consistency	$\lambda_{\text{cons}} = 0$	94.0 / 1.1	97.8 / 35.4	93.8 / 35.7
Consistency	$\boldsymbol{\lambda_{\text{cons}} = 0.20}$	**92.4 / 0.4**	**98.3 / 41.2**	**95.7 / 16.3**
Features	w/o verifier	92.4 / 1.5	62.6 / 19.8	90.6 / 2.2
Features	w/o entropy	92.3 / 0.3	62.6 / 31.8	91.8 / 12.0
Features	entropy only	96.0 / 20.4	64.6 / 2.4	87.6 / 18.7

The CVaR sweep in Appendix C.2 supports the tail-risk objective. The canonical setting $(\alpha, \varepsilon) = (0.20, 0.10)$ achieves 97.76% SR at 27.47% LLM calls. A more conservative constraint, $(0.05, 0.02)$, raises SR to 98.05% but increases escalation to 56.63%, while a relaxed setting, $(0.20, 0.15)$, reduces escalation to 21.11% at 95.29% SR. Thus, ε behaves as intended: tighter tail-risk control buys reliability through additional teacher calls.

Feature ablations show why calibrated routing needs both local quality evidence and uncertainty signals. TextWorld collapses to 62.6% SR when either verifier or entropy features are removed, and entropy-only routing stays near SLM-only performance at 64.6% SR. HumanEval+ is more forgiving: entropy-only routing reaches 96.0% SR, but at a high 20.4% escalation rate. TerminalBench also depends on richer risk features: removing verifier or entropy reduces SR to 90.6% and 91.8%, respectively.

Closed-source adaptation. R2V can also operate without token log-probabilities by replacing entropy features with verifier-distribution pseudo-entropy. This largely preserves performance on HumanEval+ and TextWorld but degrades TerminalBench; full results are provided in Appendix C.3.

R2V degrades gracefully on HumanEval+ and TextWorld under closed-source constraints. On HumanEval+, no-entropy changes SR from 93.3% to 92.8%, while pseudo-entropy slightly exceeds the full-feature result, reaching 94.2% SR with only 0.6% escalation. On TextWorld, pseudo-entropy nearly matches the full router, reaching 98.1% SR while reducing escalation from 41.7% to 37.4%, with ECE 0.054 and Brier score 0.106. TerminalBench is more sensitive: no-entropy drops from 93.3% to 90.9% SR and worsens ECE from 0.162 to 0.238, while pseudo-entropy partially recovers performance and calibration, reaching 92.8% SR at 27.1% escalation, with ECE 0.181 and Brier score 0.204. Thus, black-box deployment is practical for code and navigation tasks, while terminal-engineering agents benefit from true log-probability access when available.

6 Conclusion

We introduced **R2V-Agent** [1] , a step-level routing framework that allows an SLM agent to request help from a stronger LLM only when needed. R2V estimates failure risk from SLM uncertainty, verifier signals, and trajectory features. The SLM uses a standard BC→DPO recovery-distillation pipeline, while the main contribution is a router trained with Brier calibration and CVaR-based tail-risk control.

Across HumanEval+, TextWorld, and TerminalBench, R2V improves the reliability–cost trade-off by keeping routine steps local and escalating high-risk steps. It achieves reliability comparable to heuristic routing while using fewer teacher calls, providing a practical middle ground between low-cost but brittle SLM execution and expensive LLM-only execution.

Limitations. The router is task-family-specific. Cross-benchmark transfer fails: for example, a TextWorld-trained router achieves AUROC 0.977 in-distribution but 0.500 on HumanEval+ (Appendix C.4). This occurs because uncertainty features relate differently to failures across benchmarks, requiring per-family calibration data. Our cost model also excludes the separate costs of $K{=}5$ candidate generation, verifier scoring, and wall-clock latency. Finally, comparisons with RouterBench (Hu et al., 2024) are indirect because it studies request-level rather than step-level routing. Future work should improve verifier quality, learn more transferable risk features, and provide token-level and latency-aware cost evaluation.

[1] **Code:** Code will be open-sourced upon acceptance.

REFERENCES

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R. Featured Certification.

Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for cvar optimization in mdps, 2014. URL https://arxiv.org/abs/1406.3339.

Cognition. Introducing devin, the first ai software engineer. *Cognition Blogs*, 2024. URL https://www.cognition-labs.com/blog/introducing-devin.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025. URL https://openreview.net/forum?id=qe8BfREMrb.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024a. URL https://arxiv.org/abs/2404.14618.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024b. URL https://arxiv.org/abs/2404.14618.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. URL https://arxiv.org/abs/1706.04599.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system, 2024. URL https://arxiv.org/abs/2403.12031.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.12917.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Mike A. Merrill, Alexander G. Shaw, and Nicholas Carlini et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1357–1370. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.findings-eacl.91. URL https://aclanthology.org/2024.findings-eacl.91.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. Prompt perturbation consistency learning for robust language models. In Yvette Graham and Matthew Purver (eds.), *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1357–1370, St. Julian's, Malta, March 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-eacl.91. URL https://aclanthology.org/2024.findings-eacl.91/.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, and et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2010.03768.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021b. URL https://arxiv.org/abs/2010.03768.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition, 2018.

Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Policy gradients for CVaR-constrained policies. In *Algorithmic Learning Theory*, 2015. URL https://arxiv.org/abs/1507.07969.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024. URL https://arxiv.org/abs/2312.08935.

Zhaoyang Wang, Weilei He, Zhiyuan Liang, Xuchao Zhang, Chetan Bansal, Ying Wei, Weitong Zhang, and Huaxiu Yao. CREAM: Consistency regularized self-rewarding language models. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2410.12735.

Zhaoyang Wang, Weilei He, Zhiyuan Liang, Xuchao Zhang, Chetan Bansal, Ying Wei, Weitong Zhang, and Huaxiu Yao. Cream: Consistency regularized self-rewarding language models, 2025b. URL https://arxiv.org/abs/2410.12735.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. Agentprm: Process reward models for llm agents via step-wise promise and progress, 2025a. URL https://arxiv.org/abs/2511.08325.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. Agentprm: Process reward models for llm agents via step-wise promise and progress, 2025b. URL https://arxiv.org/abs/2511.08325.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=mXpq6ut8J3.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=mXpq6ut8J3.

yoonholee. Terminal-Bench 2.0 Trajectories. https://huggingface.co/datasets/yoonholee/terminalbench-trajectories, 2026. Dataset; Apache-2.0 license; accessed May 3, 2026.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024a. URL https://arxiv.org/abs/2307.13854.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=oKn9c6ytLx.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models, 2024. URL https://arxiv.org/abs/2410.02223.

Algorithm 1 R2V-Agent Training Pipeline

Require: Teacher LLM π_T, initial SLM π_θ, verifier V_ϕ, perturbation operators $\{\mathcal{P}_k\}$

Collect expert trajectories $\mathcal{D}_{\text{exp}}$ using π_T on training tasks
Apply $\{\mathcal{P}_k\}$ across seeds $z \in \mathcal{Z}$ to obtain perturbed trajectories $\mathcal{D}_{\text{pert}}$ and form offline trajectory pool $\mathcal{D}_{\text{traj}} \leftarrow \mathcal{D}_{\text{exp}} \cup \mathcal{D}_{\text{pert}}$
Warm-start the SLM by minimizing $\mathcal{L}_{\text{BC}}$ on $\mathcal{D}_{\text{traj}}$, yielding π_θ^{BC}
Initialize empty datasets $\mathcal{D}_{\text{pref}}, \mathcal{D}_{\text{cons}}, \mathcal{D}_{\text{route}}$
for contexts x'_t from $\mathcal{D}_{\text{traj}}$ **do**
    Sample K candidate actions $a_t^{(1:K)} \sim \pi_\theta^{\text{BC}}(\cdot \mid x'_t)$
    Score candidates with $V_\phi(x'_t, a_t^{(k)})$; optionally query teacher action $a_t^T \sim \pi_T(\cdot \mid x'_t)$
    Construct preference pairs (x'_t, a_t^+, a_t^-) and add them to $\mathcal{D}_{\text{pref}}$
    Construct paired perturbation views for consistency regularization and add them to $\mathcal{D}_{\text{cons}}$
end for
Freeze a DPO reference policy $\pi_{\text{ref}} \leftarrow \text{stopgrad}(\pi_\theta^{\text{BC}})$
Continue training the SLM from π_θ^{BC} using only $\beta_1 \mathcal{L}_{\text{DPO}} + \beta_2 \mathcal{L}_{\text{cons}}$, yielding distilled policy π_θ
for rollouts of the distilled policy π_θ under diverse perturbation seeds **do**
    At each step, sample SLM candidates, score them with V_ϕ, and extract risk features f_t
    Define episode-failure label y_t from the SLM-only continuation outcome under perturbation seed z
    Add (f_t, y_t) to $\mathcal{D}_{\text{route}}$
end for
Train router r_ψ on $\mathcal{D}_{\text{route}}$ with Brier calibration and a CVaR-constrained cost objective
return Distilled SLM π_θ, verifier V_ϕ, router r_ψ

Algorithm 2 R2V-Agent Risk-Calibrated Inference Loop

Require: Distilled SLM π_θ, Teacher LLM π_T, Verifier V_ϕ, Router r_ψ, Threshold $\tau^\star_{\text{route}}$

Initialize context $x_0 = (G, o_0)$
for $t = 0, 1, \ldots, H-1$ **do**
    Sample K candidate actions $a_t^{(1)}, \ldots, a_t^{(K)} \sim \pi_\theta(\cdot \mid x_t)$
    Score candidates $s_t^{(k)} = V_\phi(x_t, a_t^{(k)})$ and set $\hat{a}_t^{\text{SLM}} = \arg\max_k s_t^{(k)}$
    Compute risk features $f_t = \Phi(x_t, \pi_\theta, V_\phi)$
    Compute escalation probability $p_t = r_\psi(f_t)$
    Set hard deployment decision $d_t = \mathbf{1}[p_t \geq \tau^\star_{\text{route}}]$
    if $d_t = 1$ **and** Remaining Budget > 0 **then**
        Sample $a_t \sim \pi_T(\cdot \mid x_t)$
        Decrement LLM Budget
    else
        Set $a_t = \hat{a}_t^{\text{SLM}}$
    end if
    Execute a_t, observe $o_{t+1} \sim \mathcal{O}_z(\cdot \mid x_t, a_t, z)$
    Update context $x_{t+1} = (x_t, a_t, o_{t+1})$
end for

A Theoretical Guarantees for Risk-Calibrated Routing

This appendix collects the formal assumptions and proofs omitted from the main text.

We study the agent defined on the noisy-observation POMDP

$$\mathcal{M}_z = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{T}, \mathcal{O}_z, R, \gamma),$$

where $z \in \mathcal{Z}$ denotes the perturbation seed, x_t the step-t context, π_θ the distilled SLM, π_T the teacher LLM, V_ϕ the verifier, and r_ψ the router.

At each step t, the SLM generates K candidates

$$a_t^{(1)}, \dots, a_t^{(K)} \sim \pi_\theta(\cdot \mid x_t),$$

the verifier scores them by

$$s_t^{(k)} = V_\phi(x_t, a_t^{(k)}),$$

and the best SLM candidate is

$$\hat{a}_t^{\mathrm{SLM}} = \arg\max_{1 \le k \le K} V_\phi(x_t, a_t^{(k)}).$$

The router receives a feature vector f_t and outputs

$$p_t = r_\psi(f_t) \in [0, 1].$$

A hard routing decision is then obtained by thresholding:

$$d_t = \mathbf{1}[p_t \ge \tau_{\mathrm{route}}].$$

Throughout, $y_t \in \{0, 1\}$ denotes the episode-derived SLM failure label attached to step t. For a logged SLM rollout $\tau^{\mathrm{SLM}}(z)$, let $S_{\mathrm{SLM}}(z) \in \{0, 1\}$ be its binary episode success. Each context $x_t \in \tau^{\mathrm{SLM}}(z)$ receives $y_t = \mathbf{1}[S_{\mathrm{SLM}}(z) = 0]$. This label is not a per-step correctness annotation and is not defined by comparing the verifier score of the SLM action against a teacher action. The teacher enters the framework as the escalation policy used by the routed system; the calibration target for the router is the conditional probability that the deployed SLM will fail the episode from the current local evidence.

A.1 Assumptions

Assumption 1 (Boundedness and measurability)**.** For every step t,

$$V_\phi(x_t, a_t) \in [0, 1], \quad r_\psi(f_t) \in [0, 1], \quad y_t \in \{0, 1\}, \quad S_z \in \{0, 1\}.$$

Moreover, the feature vector satisfies $\|f_t\|_2 \le B_f$ for some finite constant $B_f > 0$.

Assumption 2 (Well-defined calibration target)**.** Define the conditional escalation probability

$$q^\star(f) := \mathbb{P}(y_t = 1 \mid f_t = f).$$

We assume that $q^\star(f)$ is well-defined for almost every f.

Assumption 3 (Cost-sensitive routing surrogate)**.** For hard routing analysis, we evaluate a routing decision $d_t \in \{0, 1\}$ with the stepwise surrogate loss

$$\ell_{\mathrm{route}}(d_t, y_t) = c_{\mathrm{SLM}}(1 - d_t) + c_{\mathrm{LLM}} d_t + \kappa\, y_t(1 - d_t),$$

where $c_{\mathrm{SLM}} < c_{\mathrm{LLM}}$ and $\kappa > 0$ is the penalty for not escalating on a step where escalation was actually needed.

Assumption 4 (Perturbation pairing and stepwise Lipschitzness)**.** Suppose two perturbation seeds $z, z' \in \mathcal{Z}$ produce two observed contexts $x_t^{(z)}$ and $x_t^{(z')}$ corresponding to the same latent environment state s_t. Let

$$P_t^{(z)} := \pi_\theta(\cdot \mid x_t^{(z)}), \qquad P_t^{(z')} := \pi_\theta(\cdot \mid x_t^{(z')}).$$

Assume the one-step surrogate loss $\ell_t(P; s_t) \in [0, 1]$ satisfies

$$|\ell_t(P; s_t) - \ell_t(Q; s_t)| \le L\ \mathrm{TV}(P, Q)$$

for all action distributions P, Q and some constant $L > 0$.

A.2 Calibration and Bayes-optimal routing

Theorem 5 (Brier-optimal calibration of the router)**.** Under Assumptions 1 and 2, the population minimizer of the Brier risk

$$\mathcal{R}_{\mathrm{Brier}}(r_\psi) = \mathbb{E}\big[(r_\psi(f_t) - y_t)^2\big]$$

is the conditional escalation probability

$$r^\star(f) = q^\star(f) = \mathbb{P}(y_t = 1 \mid f_t = f).$$

Proof. Condition on $f_t = f$, and write $p = r_\psi(f)$ and $q = q^\star(f) = \mathbb{P}(y_t = 1 \mid f_t = f)$. Since $y_t \in \{0, 1\}$,

$$\begin{aligned}\mathbb{E}\big[(p - y_t)^2 \mid f_t = f\big] &= q(p-1)^2 + (1-q)p^2 \\ &= (p-q)^2 + q(1-q).\end{aligned}$$

The second term is constant in p, and the first is uniquely minimized at $p = q$. Taking expectation over f_t proves the claim. □

Corollary 6 (Cost-optimal hard routing threshold)**.** Under Assumptions 1, 2, and 3, the Bayes-optimal hard routing rule for the unconstrained one-step surrogate ℓ_{route} is

$$d_t^\star = \mathbf{1}\left[q^\star(f_t) \geq \tau_{\mathrm{route}}^\star\right],$$

where

$$\tau_{\mathrm{route}}^\star = \min\left\{1, \max\left\{0, \frac{c_{\mathrm{LLM}} - c_{\mathrm{SLM}}}{\kappa}\right\}\right\}.$$

If $0 \leq c_{\mathrm{LLM}} - c_{\mathrm{SLM}} \leq \kappa$, the clipped threshold equals $(c_{\mathrm{LLM}} - c_{\mathrm{SLM}})/\kappa$.

Proof. Fix $f_t = f$, and let $q = q^\star(f)$. If $d_t = 1$, then

$$\mathbb{E}[\ell_{\mathrm{route}}(1, y_t) \mid f_t = f] = c_{\mathrm{LLM}}.$$

If $d_t = 0$, then

$$\mathbb{E}[\ell_{\mathrm{route}}(0, y_t) \mid f_t = f] = c_{\mathrm{SLM}} + \kappa q.$$

Escalation is optimal iff $c_{\mathrm{LLM}} \leq c_{\mathrm{SLM}} + \kappa q$, equivalently

$$q \geq \frac{c_{\mathrm{LLM}} - c_{\mathrm{SLM}}}{\kappa}.$$

Because $q \in [0, 1]$, thresholds outside $[0, 1]$ are equivalent to the nearest feasible endpoint, yielding the clipped expression. □

Theorem 7 (Regret bound from router miscalibration)**.** Let

$$\begin{aligned}\hat{d}_t &= \mathbf{1}\left[r_\psi(f_t) \geq \tau_{\mathrm{route}}^\star\right], \\ \tau_{\mathrm{route}}^\star &= \min\left\{1, \max\left\{0, \frac{c_{\mathrm{LLM}} - c_{\mathrm{SLM}}}{\kappa}\right\}\right\},\end{aligned}$$

be the plug-in hard router induced by a probabilistic router r_ψ, and let $d_t^\star$ denote the Bayes-optimal rule from Corollary 6. Then

$$\mathbb{E}\Big[\ell_{\mathrm{route}}(\hat{d}_t, y_t) - \ell_{\mathrm{route}}(d_t^\star, y_t)\Big] \leq \kappa\, \mathbb{E}[|r_\psi(f_t) - q^\star(f_t)|].$$

Proof. Condition on $f_t = f$, and write

$$q = q^\star(f), \qquad \widehat{q} = r_\psi(f), \qquad \tau^\star = \tau_{\mathrm{route}}^\star$$

From the proof of Corollary 6,

$$R_f(1) = c_{\mathrm{LLM}}, \qquad R_f(0) = c_{\mathrm{SLM}} + \kappa q,$$

so

$$|R_f(0) - R_f(1)| = \kappa|q - \tau^\star|.$$

If $\hat{d}(f) \neq d^\star(f)$, then q and $\widehat{q}$ lie on opposite sides of $\tau^\star$, hence

$$|q - \tau^\star| \leq |\widehat{q} - q|.$$

Therefore the conditional regret is bounded by

$$\mathbb{E}\big[\ell_{\text{route}}(\hat{d}_t, y_t) - \ell_{\text{route}}(d_t^\star, y_t) \,\big|\, f_t = f\big] \leq \kappa|\widehat{q} - q|.$$

Taking expectation over f_t yields the result. □

A.3 Consistency regularization and perturbation transfer

Lemma 8 (Total variation is controlled by Jensen–Shannon divergence)**.** For any two distributions P and Q,

$$\mathrm{TV}(P, Q) \leq \sqrt{2\ \mathrm{JSD}(P\|Q)}.$$

Proof. Let $M = (P + Q)/2$. By the triangle inequality,

$$\mathrm{TV}(P, Q) \leq \mathrm{TV}(P, M) + \mathrm{TV}(Q, M).$$

Applying Pinsker's inequality to each term gives

$$\mathrm{TV}(P, M) \leq \sqrt{\tfrac{1}{2}\,\mathrm{KL}(P\|M)},$$
$$\mathrm{TV}(Q, M) \leq \sqrt{\tfrac{1}{2}\,\mathrm{KL}(Q\|M)}.$$

Using $\sqrt{a} + \sqrt{b} \leq \sqrt{2(a + b)}$ for $a, b \geq 0$, we obtain

$$\begin{aligned}\mathrm{TV}(P, Q) &\leq \sqrt{2\left(\tfrac{1}{2}\,\mathrm{KL}(P\|M) + \tfrac{1}{2}\,\mathrm{KL}(Q\|M)\right)} \\ &= \sqrt{2\ \mathrm{JSD}(P\|Q)}.\end{aligned}$$

□

Theorem 9 (Perturbation-generalization bound from consistency regularization)**.** Assume Assumptions 1 and 4. For a perturbation seed z, define the episodic surrogate risk

$$R_z(\pi_\theta) := \mathbb{E}\left[\sum_{t=1}^{H} \ell_t\Big(\pi_\theta(\cdot \mid x_t^{(z)}); s_t\Big)\right].$$

Then for any two perturbation seeds $z, z' \in \mathcal{Z}$,

$$|R_z(\pi_\theta) - R_{z'}(\pi_\theta)| \leq HL\sqrt{2\,\mathcal{L}_{\text{cons}}(\theta)},$$

where

$$\mathcal{L}_{\text{cons}}(\theta) = \mathbb{E}_{t,z,z'}\left[\mathrm{JSD}\Big(\pi_\theta(\cdot \mid x_t^{(z)}) \,\Big\|\, \pi_\theta(\cdot \mid x_t^{(z')})\Big)\right].$$

Consequently, if a training seed z_{train} is representative of the seen perturbations, then for any unseen test seed z_{test},

$$R_{z_{\text{test}}}(\pi_\theta) \leq R_{z_{\text{train}}}(\pi_\theta) + HL\sqrt{2\,\mathcal{L}_{\text{cons}}(\theta)}.$$

Proof. Write

$$R_z(\pi_\theta) - R_{z'}(\pi_\theta) = \mathbb{E}\left[\sum_{t=1}^{H}\Big(\ell_t(P_t^{(z)}; s_t) - \ell_t(P_t^{(z')}; s_t)\Big)\right].$$

By Assumption 4,

$$\Big|\ell_t(P_t^{(z)}; s_t) - \ell_t(P_t^{(z')}; s_t)\Big| \leq L\ \mathrm{TV}\Big(P_t^{(z)}, P_t^{(z')}\Big).$$

Applying Lemma 8,

$$\mathrm{TV}\Big(P_t^{(z)}, P_t^{(z')}\Big) \le \sqrt{2\ \mathrm{JSD}\Big(P_t^{(z)}\Big\|P_t^{(z')}\Big)}.$$

Using Jensen's inequality over the expectation defining $\mathcal{L}_{\mathrm{cons}}(\theta)$, each stepwise term is bounded by $\sqrt{2\,\mathcal{L}_{\mathrm{cons}}(\theta)}$. Summing over H steps yields

$$|R_z(\pi_\theta) - R_{z'}(\pi_\theta)| \le HL\sqrt{2\,\mathcal{L}_{\mathrm{cons}}(\theta)}.$$

Setting $z = z_{\mathrm{test}}$ and $z' = z_{\mathrm{train}}$ gives the second claim. □

Remark 1. Theorem 9 is stated for an episodic surrogate risk that decomposes over steps. Whenever the binary episode failure $1 - S_z$ is upper bounded by such a surrogate, the same control transfers directly to the robust routing objective in the main text.

A.4 Auxiliary results for verifier-guided distillation

Definition 1 (Good candidate set)**.** For a fixed context x_t and threshold $\gamma \in [0, 1]$, define

$$\mathcal{G}_\gamma(x_t) := \{a \in \mathcal{A} : V_\phi(x_t, a) \ge \gamma\}.$$

Also define the SLM coverage probability

$$\mu_\gamma(x_t) := \mathbb{P}_{a\sim\pi_\theta(\cdot|x_t)}\big(a \in \mathcal{G}_\gamma(x_t)\big).$$

Proposition 10 (Best-of-K verifier selection improves with coverage)**.** Fix a context x_t, and suppose K candidate actions are sampled i.i.d. from $\pi_\theta(\cdot \mid x_t)$. Assume that for every good action $a^+ \in \mathcal{G}_\gamma(x_t)$ and every bad action $a^- \notin \mathcal{G}_\gamma(x_t)$, the verifier misranks the pair with probability at most η_V, i.e.,

$$\mathbb{P}\big(V_\phi(x_t, a^-) \ge V_\phi(x_t, a^+)\big) \le \eta_V.$$

Then

$$\mathbb{P}\big(\hat{a}_t^{\mathrm{SLM}} \in \mathcal{G}_\gamma(x_t)\big) \ge 1 - (1 - \mu_\gamma(x_t))^K - \frac{K(K-1)}{2}\eta_V.$$

Proof. Let $E_{\mathrm{good}} = \{\hat{a}_t^{\mathrm{SLM}} \in \mathcal{G}_\gamma(x_t)\}$, and write its complement as the union of two events: (i) no good candidate is sampled; or (ii) at least one good candidate is sampled, but a bad candidate is selected due to verifier misranking. The first event has probability

$$(1 - \mu_\gamma(x_t))^K$$

by independence. For the second event, there are at most $K(K-1)/2$ good–bad pairs among the K samples, and each can be misranked with probability at most η_V; a union bound therefore gives

$$\mathbb{P}(E_{\mathrm{bad}}) \le (1 - \mu_\gamma(x_t))^K + \frac{K(K-1)}{2}\eta_V.$$

Taking complements yields the claim. □

Proposition 11 (Pairwise sign consistency of verifier-noisy DPO)**.** Consider a fixed context x_t and two actions a^+, a^-. Let the true pairwise preference label be

$$y^\star \in \{+1, -1\},$$

where $y^\star = +1$ means a^+ is truly preferred to a^-. Suppose the verifier produces a noisy pairwise label $\widetilde{y}$ such that

$$\mathbb{P}(\widetilde{y} = y^\star \mid x_t, a^+, a^-) = 1 - \eta,$$
$$\mathbb{P}(\widetilde{y} = -y^\star \mid x_t, a^+, a^-) = \eta,$$

with $\eta < 1/2$.

Let the DPO score difference be

$$u_\theta(x_t, a^+, a^-) = \beta\left[\log\frac{\pi_\theta(a^+ \mid x_t)}{\pi_{\mathrm{ref}}(a^+ \mid x_t)} - \log\frac{\pi_\theta(a^- \mid x_t)}{\pi_{\mathrm{ref}}(a^- \mid x_t)}\right].$$

Then the population minimizer of the noisy logistic loss

$$\mathcal{L}_{\mathrm{DPO,noisy}}(\theta) = \mathbb{E}\Big[\log(1 + \exp(-\widetilde{y}\,u_\theta))\Big]$$

has the same sign as the true preference:

$$\mathrm{sign}\big(u_\theta^\star(x_t, a^+, a^-)\big) = y^\star.$$

Proof. It suffices to consider the case $y^\star = +1$, since $y^\star = -1$ is symmetric. Conditional on (x_t, a^+, a^-), the noisy label equals $+1$ with probability $1-\eta$ and -1 with probability η. Hence the scalar conditional risk is

$$R(u) = (1-\eta)\log(1+e^{-u}) + \eta\log(1+e^{u}).$$

Differentiating gives

$$R'(u) = -\frac{1-\eta}{1+e^{u}} + \frac{\eta e^{u}}{1+e^{u}} = \frac{\eta e^{u} - (1-\eta)}{1+e^{u}}.$$

The stationary point satisfies

$$e^{u} = \frac{1-\eta}{\eta}.$$

Since $\eta < 1/2$, we have $(1-\eta)/\eta > 1$, so the minimizer $u^\star = \log((1-\eta)/\eta)$ is strictly positive. Therefore $\operatorname{sign}(u^\star) = +1 = y^\star$. The symmetric argument yields the result for $y^\star = -1$. □

B Additional Experimental Details

B.1 Detailed Dataset Construction

All benchmarks use a task-level $70/15/15$ train/validation/test split with seed 42. Each clean trajectory is replayed under 5 independently sampled perturbation seeds to produce the noisy training and evaluation distributions.

HumanEval+. We use the full EvalPlus benchmark (Liu et al., 2023), comprising 164 Python programming problems. Each problem is presented as a function signature with a docstring. The agent interacts with a three-action environment: `write_code`, `test`, and `submit`. Clean teacher trajectories are collected with Gemini 3 Flash Preview, yielding 820 trajectories, i.e., 5 per task. Perturbed variants are generated by applying all four perturbation operators simultaneously through $\mathcal{P}_{\text{composite}}$: tool flakiness injects HTTP errors, stale cache markers, and truncated responses into test-runner output; partial observability drops or reorders log lines; prompt injection embeds adversarial directives in problem docstrings or test outputs; and distractors append plausible but incorrect code snippets and red-herring error messages to observations. The executor provides a deterministic binary success signal by running the full EvalPlus test suite.

TextWorld. We use ALFWorld (Shridhar et al., 2021b), a suite of interactive text-based household tasks. The benchmark comprises 250 tasks split across six task types: pick-and-place, pick-clean-then-place, pick-heat-then-place, pick-cool-then-place, look-at-object-in-light, and pick-two-objects. The agent issues text commands and receives textual observations describing the current room, visible objects, and inventory. Teacher trajectories are collected with Gemini 3 Flash Preview, yielding $800/200/250$ train/validation/test episodes. All four perturbation types are applied at training and test time.

TerminalBench. We use TerminalBench (Merrill et al., 2026), a benchmark of 89 real-world terminal engineering tasks sourced from the `yoonholee/terminalbench-trajectories` HuggingFace dataset. Tasks span systems-programming scenarios such as compiling language toolchains, repairing git histories, building gRPC services, and optimizing numerical kernels. Each task is graded by automated test suites embedded in the benchmark. Teacher trajectories are collected from a diverse pool of frontier models, including Claude Opus 4.6, GPT-5.5, Gemini 3.1 Pro, and others. The split yields approximately $62/13/14$ train/validation/test tasks with $3{,}750$ episodes in total: $2{,}625$ clean and $1{,}125$ composite-perturbed. Unlike the other benchmarks, TerminalBench perturbations combine only tool flakiness and partial observability.

B.2 Verifier Details

HumanEval verifier. The HumanEval+ verifier decomposes each score into execution and structural signals. The primary signal, with weight 0.35, runs the candidate code in an isolated subprocess against a smoke-test harness. The remaining weight is distributed across structural checks: required function definition (0.20), non-trivial control flow (0.15), non-trivial return statement (0.10), import

sanity (0.10), and length regularity (0.10). Two multiplicative penalties are then applied: a reward-hacking penalty for hardcoded lookup tables, bare literal returns, and exception-swallowing patterns; and a repetition penalty for re-submitting code identical to a previously failed attempt. A $0.85\times$ discount is applied when prompt-injection markers are detected. When a previous test result is visible, the final score blends the structural estimate and observed test outcome using a $60/40$ split. On the held-out test split, the verifier achieves episode-level AUROC 0.856 and Brier score 0.172.

TextWorld verifier. The TextWorld verifier combines five local signals: previous observation quality (0.25), action-type base score (0.25), goal alignment (0.25), non-repetition (0.15), and non-oscillation (0.10). Observation quality parses the most recent environment response for progress indicators such as *you take*, *you put*, and *you go to*, versus failure strings such as *that action did not help*. Action-type priors score task-progress verbs such as *take*, *put*, *clean*, and *heat* above navigational or idle actions. Goal alignment measures token overlap between action arguments and key nouns extracted from the task goal. When a scalar reward is visible, the weighted score is blended with the observed reward using a $50/50$ split. On the held-out TextWorld split, mean-aggregated verifier scores achieve episode-level AUROC ≈ 0.78, indicating moderate but imperfect local discrimination. This supports our design choice to use the verifier as one input to the learned router rather than as an oracle routing rule.

TerminalBench verifier. The TerminalBench verifier operates without Docker or subprocess execution. It scores each action by parsing terminal output already present in the trajectory context. For `run [cmd]` actions, it extracts the most recent "New Terminal Output:" block, strips perturbation noise, and scans for explicit success or failure signals. Success patterns such as `PASSED`, "all tests passed", "build success", and "wrote N bytes" yield terminal scores in $[0.70, 1.00]$; failure patterns such as "command not found", `Traceback`, non-zero exit codes, and segmentation faults yield scores in $[0.08, 0.18]$. Commands are also categorized as *eval*, *build*, *run*, *install*, *write*, or *explore*, with category-specific base scores. The final step score blends terminal evidence and command category, with evaluation commands receiving the privileged score $\mathrm{clamp}(0.15 + 0.85 s_{\mathrm{terminal}})$. Repetition and adversarial-injection penalties are applied multiplicatively.

For `mark_task_complete`, the verifier searches the last $3{,}000$ characters for terminal success evidence and returns $\mathrm{clamp}(0.20 + 0.75 s_{\mathrm{terminal}}, 0.20, 0.95)$ when such evidence exists. A clean bash prompt yields 0.68, no signal yields 0.55, `image_read` returns 0.52, and unrecognized actions return 0.40. On the held-out test split, the verifier achieves episode-level AUROC 0.508 and Brier score 0.276, reflecting near-chance separation when local step scores are mean-aggregated. This is expected: individual step scores encode local command quality rather than complete episode outcome, and the router learns to bridge this local-to-global gap.

Step labels. The verifier V_ϕ scores each step using local signals, while the router label y_t is global but attached to the current context: it equals 1 when the fixed SLM rollout containing that context ends in episode failure, and 0 when the rollout succeeds. We do not observe ground-truth step success labels, and we do not define y_t by evaluating whether a teacher action would have prevented the failure. The router therefore learns a local-to-global mapping: from current SLM uncertainty, verifier statistics, and execution context to the probability that continuing with the SLM will fail the episode.

Table 4: **Step-label distribution and verifier discrimination.** "Successful step" means the current context belongs to an SLM-only rollout whose binary episode outcome is success; "failed step" means it belongs to an SLM-only rollout whose binary episode outcome is failure.

Benchmark	Succ.steps	Fail.steps	Score(succ.)	Score(fail.)	AUROC(ep.)
HumanEval+	91.89%	8.11%	0.620	0.336	0.856
TextWorld	64.60%	35.40%	$\approx$0.63	$\approx$0.51	$\approx$0.78
TerminalBench	55.55%	44.45%	0.406	0.398	0.508

Table 5: **R2V-Agent per-model results** under noisy evaluation with 95% bootstrap confidence intervals. Oracle is shown as a non-deployable hindsight reference.

Benchmark	Model	SR (%)	95% CI	LLM%
HumanEval+	Gemma-9B	91.9	[89.6, 93.8]	0.50%
	LLaMA-3.1-8B	**95.8**	**[94.3, 97.3]**	**1.46%**
	Qwen2.5-14B	93.6	[91.1, 94.9]	0.25%
	Qwen2.5-7B	92.9	[90.1, 94.2]	0.36%
	Oracle	*98.6*	–	*8.1%*
TextWorld	Gemma-9B	98.0	[96.4, 98.8]	38.3%
	LLaMA-3.1-8B	**98.4**	**[97.0, 99.0]**	**48.8%**
	Qwen2.5-14B	98.1	[96.6, 98.9]	38.4%
	Qwen2.5-7B	98.3	[96.9, 99.0]	41.2%
	Oracle	*98.6*	–	*35.4%*
TerminalBench	Gemma-9B	92.8	[90.6, 94.7]	38.9%
	LLaMA-3.1-8B	95.5	[93.8, 97.0]	67.2%
	Qwen2.5-14B	89.2	[86.1, 92.0]	13.2%
	Qwen2.5-7B	**95.7**	**[93.9, 97.1]**	**16.3%**
	Oracle	*97.8*	–	*18.4%*

B.3 Router Architecture and Training

The router $r_\psi : \mathbb{R}^{15} \to [0, 1]$ is a shallow MLP with hidden widths 128 and 64, GELU activations, batch normalization, dropout $p = 0.2$, and a sigmoid output. Learnable post-hoc temperature scaling (Guo et al., 2017) is applied to the output logits. The router has approximately 10,000 parameters and runs entirely on CPU.

At each step t, the distilled SLM samples $K = 5$ candidate actions $a_t^{(1)}, \ldots, a_t^{(K)}$ with vLLM (Kwon et al., 2023). The verifier scores all candidates, and the resulting 15-dimensional feature vector f_t contains token-level entropy and log-probability statistics; verifier score mean, standard deviation, spread, best score, and worst score; candidate consistency and semantic entropy (Kuhn et al., 2023); horizon fraction and absolute step index; normalized context length; and a goal-length proxy. Benchmark and perturbation-type one-hot features are reserved but disabled in the reported experiments.

Router training minimizes the differentiable CVaR-constrained Lagrangian in Eq. equation 9, using $p_t = r_\psi(f_t)$ as the soft routing decision during optimization. Hard thresholded decisions are used only for validation-time threshold selection and test-time execution. The primal optimizer is AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-3} and weight decay 10^{-4}. A separate Adam optimizer with learning rate 10^{-2} performs gradient ascent on the log-multiplier $\log \lambda$. We use $\alpha = 0.20$, $\epsilon = 0.10$, Brier calibration weight 1.0, and cost ratio $c_{\text{LLM}}/c_{\text{SLM}} = 50$. The router is trained for 20 epochs with cosine learning-rate annealing on batches of 4,096 steps.

C Additional Results

C.1 Per-Model Results

Table 5 disaggregates R2V performance across the four SLM backbones. On HumanEval+, every backbone routes fewer than 2% of steps to the teacher, confirming that the benchmark is a low-escalation regime. On TextWorld, all backbones reach high SR, between 98.0% and 98.4%, at escalation rates of 38.3–48.8%, close to the oracle's 35.4% escalation rate. TerminalBench is more uneven: Qwen2.5-7B reaches 95.7% SR with only 16.3% escalation, while LLaMA-3.1-8B reaches 95.5% SR but requires 67.2% escalation. This confirms that R2V's benefit depends on both the SLM backbone and the identifiability of residual risk from online features.

Table 6: **Worst-seed success rate (%)**: the lowest per-seed success rate under perturbation, averaged over four SLM backbones. Higher is better. Mean success rate is reported in Table 1.

Method	HumanEval+	TextWorld	TerminalBench
SLM-only	91.9	64.6	80.6
Entropy Router	92.9	64.6	86.9
Heuristic Router	97.4	98.0	94.5
R2V (ours)	**93.2**	**98.0**	**92.8**
LLM-only	98.0	98.0	98.0

C.2 Full Ablations

Worst-seed robustness. We report the lowest per-seed success rate under the evaluation perturbations to measure whether routing remains reliable on the most difficult seeds.

R2V's worst-seed reliability remains close to its mean success rate; for example, TerminalBench reaches 92.8% worst-seed SR compared with 93.3% mean SR. R2V also improves worst-seed SR over SLM-only on every benchmark, including an increase from 80.6% to 92.8% on TerminalBench. This is the tail behavior targeted by the CVaR constraint: improving reliability on difficult perturbation seeds without the heuristic router's substantially higher escalation rate.

Consistency regularization. Table 7 sweeps $\lambda_{\text{cons}} \in \{0, 0.05, 0.2, 0.5, 1.0\}$ on Qwen2.5-7B. HumanEval+ exposes a cost–accuracy tradeoff: $\lambda_{\text{cons}} = 0$ gives the highest SR at 94.0%, while larger values reduce escalation, reaching 0.2% at $\lambda_{\text{cons}} = 1.0$. TextWorld is comparatively insensitive, staying between 97.6% and 98.3% SR across the sweep. TerminalBench benefits from moderate consistency: $\lambda_{\text{cons}} = 0.20$ gives the best SR, 95.7%, while reducing LLM calls from 35.7% to 16.3%. Increasing λ_{cons} further reduces escalation slightly, but no longer improves SR.

Table 7: **Effect of consistency regularization** λ_{cons} on Qwen2.5-7B under noisy evaluation. $\lambda_{\text{cons}} = 0$ corresponds to pure DPO without the JSD term.

	HumanEval+		TextWorld		TerminalBench	
λ_{cons}	SR (%)	LLM%	SR (%)	LLM%	SR (%)	LLM%
0	94.0	1.1%	97.8	35.4%	93.8	35.7%
0.05	93.4	0.8%	98.0	38.2%	94.9	24.0%
0.20	**92.4**	**0.4%**	**98.3**	**41.2%**	**95.7**	**16.3%**
0.50	92.6	0.3%	98.1	43.0%	95.3	15.9%
1.0	92.1	0.2%	97.6	45.0%	94.7	15.0%

CVaR sensitivity. Table 8 samples from a 5×6 grid of (α, ε) configurations evaluated on the combined benchmark suite. The sweep supports the role of the CVaR constraint in the routing objective. Tighter tail-risk budgets increase teacher usage and improve reliability: $(\alpha, \varepsilon) = (0.05, 0.02)$ reaches 98.05% SR but escalates 56.63% of steps. Relaxing the constraint reduces cost with a measurable SR tradeoff: $(0.20, 0.15)$ uses 21.11% LLM calls and reaches 95.29% SR. The canonical setting $(0.20, 0.10)$ lies between these extremes, with 97.76% SR and 27.47% LLM calls.

Feature groups. Table 9 evaluates progressively masked feature dimensions at inference time without retraining. The results explain why the learned router outperforms simple entropy routing. On TextWorld, entropy-only routing reaches only 64.6% SR, matching the SLM-only baseline, while verifier-only routing recovers to 97.9% SR but requires 78.5% LLM calls. The full feature set gives a better balance, reaching 98.3% SR with 41.2% escalation. On HumanEval+, entropy is more useful, but entropy-only routing still over-escalates at 20.4% LLM calls. On TerminalBench, feature removal mainly hurts the router's ability to detect hard shell states: entropy-only falls to 87.6% SR, and removing the verifier lowers SR to 90.6%.

Table 8: **CVaR hyperparameter sensitivity** on combined HumanEval+/TextWorld/TerminalBench, averaged over all SLM backbones. Bold indicates the canonical operating point.

α	ε	SR (%)	LLM%
0.05	0.02	98.05	56.63
0.05	0.10	97.38	27.75
0.10	0.02	97.28	48.59
0.10	0.10	97.89	26.23
0.20	**0.10**	**97.76**	**27.47**
0.20	0.15	95.29	21.11
0.50	0.10	97.82	29.61
0.50	0.15	95.55	21.49

Table 9: **Feature group ablation** on Qwen2.5-7B under noisy evaluation. Feature masking is applied at inference time without retraining.

	HumanEval+		**TextWorld**		**TerminalBench**	
Feature set	SR	LLM%	SR	LLM%	SR	LLM%
All features	**92.4**	**0.4%**	**98.3**	**41.2%**	**95.7**	**16.3%**
Verifier + entropy	96.8	9.7%	98.4	82.0%	95.1	24.0%
Verifier only	94.4	6.3%	97.9	78.5%	94.8	20.7%
Entropy only	96.0	20.4%	64.6	2.4%	87.6	18.7%
Log-prob only	94.1	1.5%	96.5	64.3%	91.0	5.4%
Step/context only	93.2	2.8%	70.4	1.2%	86.8	1.8%
w/o verifier	92.4	1.5%	62.6	19.8%	90.6	2.2%
w/o entropy	92.3	0.3%	62.6	31.8%	91.8	12.0%

C.3 Closed-Source Model Adaptation

Commercial SLM APIs may hide token log-probabilities, making entropy-based router features unavailable. We therefore evaluate two black-box variants: **no entropy**, which zeros the policy-uncertainty dimensions and retains verifier, candidate-diversity, and context features; and **pseudo-entropy**, which replaces token entropy with normalized entropy over the verifier-score distribution. For verifier scores $s_k = V_\phi(x_t, a_t^{(k)})$ over the K sampled candidates, we define

$$\begin{aligned} \tilde{p}_k &= \frac{\exp(s_k)}{\sum_{j=1}^{K} \exp(s_j)}, \\ H_{\text{ver}}(x_t) &= -\frac{1}{\log K} \sum_{k=1}^{K} \tilde{p}_k \log \tilde{p}_k. \end{aligned} \tag{13}$$

This provides a black-box uncertainty proxy without requiring SLM token log-probabilities.

With verifier-distribution pseudo-entropy, R2V achieves 94.2% success with 0.60% escalation on HumanEval+, 98.1% success with 37.4% escalation on TextWorld, and 92.8% success with 27.1% escalation on TerminalBench. These results indicate that the black-box uncertainty proxy largely preserves performance on HumanEval+ and TextWorld, while TerminalBench remains more sensitive to the absence of token log-probabilities.

Table 10: **Closed-source adaptation results** averaged across four SLM backbones under noisy evaluation. "Full" is the canonical 15-dimensional feature set. ECE and Brier score are measured on the router validation split.

Benchmark	Feature variant	SR (%)	LLM%	ECE	Brier
HumanEval+	Full	93.3	0.65%	0.067	0.114
	No entropy	92.8	0.50%	0.074	0.122
	Pseudo-entropy	**94.2**	**0.60%**	**0.069**	**0.116**
TextWorld	Full	98.2	41.7%	0.052	0.102
	No entropy	97.4	37.0%	0.065	0.124
	Pseudo-entropy	**98.1**	**37.4%**	**0.054**	**0.106**
TerminalBench	Full	93.3	33.9%	0.162	0.185
	No entropy	90.9	25.6%	0.238	0.262
	Pseudo-entropy	**92.8**	**27.1%**	**0.181**	**0.204**

C.4 Out-of-Distribution Transfer and Query-Level Comparison

This section examines how the router generalizes across benchmarks and how step-level routing compares to a query-level rule. Two caveats apply throughout. First, these analyses use per-benchmark ablation router checkpoints and an offline evaluation surrogate that credits an episode with success if at least one escalation occurs on it; absolute success rates are therefore not directly comparable to Table 1, and the comparisons should be read relatively, holding the SLM, verifier, cost model, and evaluation protocol fixed across the routing rules. Second, this analysis covers HumanEval+ and TextWorld; TerminalBench trajectories were unavailable for reconstruction.

Leave-one-benchmark-out transfer. The 15-dimensional router feature vector contains no benchmark or perturbation identity, so a trained router can be applied unchanged to a held-out benchmark. We measure the AUROC of the predicted risk score against ground-truth SLM failure (0.5 is chance); AUROC is threshold-free and is therefore unaffected by the near-bimodal router outputs. Table 11 shows that transfer fails: a TextWorld-trained router reaches AUROC 0.977 in-distribution but exactly 0.500 on HumanEval+, where it emits a near-constant score. Pooling training data across benchmarks does not help; it lowers the TextWorld AUROC from 0.977 to 0.691, because signal from different benchmarks interferes. The reverse direction, training on HumanEval+ and evaluating on TextWorld, spans chance in both directions across five checkpoints, consistent with noise.

Table 11: **Leave-one-benchmark-out transfer.** AUROC of predicted risk against ground-truth SLM failure (0.5 = chance). A router trained on one benchmark does not transfer to a held-out one.

Trained on	Evaluated on	AUROC
TextWorld	TextWorld (in-distribution)	0.977
TextWorld	HumanEval+ (unseen)	0.500
Pooled (all benchmarks)	TextWorld	0.691

The transfer signal is genuine, not a length artifact. Failed TextWorld episodes are longer than successful ones (24.3 versus 5.2 steps), and because the SLM-failure label is episode-level, step index is correlated with the label. To rule out that the router merely exploits episode length, we restrict evaluation to step 0, where the step-index feature carries no information (its own AUROC is 0.500). The TextWorld router still scores 0.983 at step 0 and 0.986 over steps 0–2, confirming that its discrimination does not come from position in the episode. The same stratification on HumanEval+ leaves the router at chance, so its weakness there is likewise not a length artifact.

Why transfer fails: the feature–risk relationship inverts. Table 12 reports the single-feature AUROC of each uncertainty signal against SLM failure on the two benchmarks. Several features point in opposite directions: predictive entropy predicts failure on HumanEval+ (0.652) but the opposite on TextWorld (0.410), and mean candidate log-probability runs 0.401 versus 0.658. A router fit on one benchmark is therefore anti-correlated on the other, which explains both the failed transfer and the loss from pooling.

Table 12: **Single-feature AUROC against SLM failure.** Several uncertainty features invert their relationship with risk across benchmarks.

Feature	HumanEval+	TextWorld
Predictive entropy	0.652	0.410
Best candidate log-prob	0.401	0.658
Mean verifier score	0.432	0.340
Semantic entropy	0.591	0.448

Query-level versus step-level routing. A direct comparison to query-level routers such as RouterBench is ill-posed, because those score precomputed responses at the level of a whole query, with no environment, intermediate steps, or perturbations, so a step-level router has nothing to route within. Instead we implement the routing *rule* they use — a single escalation decision per task, made before execution — inside our own setting, holding the SLM, verifier, cost model, and evaluation fixed. Table 13 reports the result. On TextWorld, step-level routing recovers 12.3 points of success while escalating only 2.4% of steps, precisely because the decision can be deferred until evidence of trouble appears; at a high escalation budget it is also slightly cheaper. On HumanEval+ the advantage is not demonstrable, because the router's scores are near-degenerate there and the two rules coincide at every threshold.

Table 13: **Query-level versus step-level routing** under the offline surrogate, holding the SLM, verifier, and budget fixed. Success rates are relative, not end-to-end (see caveats at the start of this section).

Benchmark	Routing	Threshold	Success	LLM calls
TextWorld	Query-level	0.6	0.640	0.0%
TextWorld	Step-level (ours)	0.6	0.763	2.4%
TextWorld	Query-level	0.1	0.995	63.5%
TextWorld	Step-level (ours)	0.1	0.999	62.3%
HumanEval+	Query-level	0.2–0.9	0.928	0.0%
HumanEval+	Step-level (ours)	0.2–0.9	0.928	0.0%

D Additional Related Work

LLM cascades and model routing. A large body of recent work studies how to reduce LLM inference cost by routing requests across models of different capability and price. Cascade-style methods query a cheaper model first and escalate only when a scoring rule predicts that the response is insufficient (Chen et al., 2024; Dekoninck et al., 2025). Ex-ante routers instead predict the appropriate model before generation, often using query embeddings, preference labels, or correctness labels (Ong et al., 2025; Ding et al., 2024a; Zhuang et al., 2024; Hu et al., 2024). These approaches are well matched to settings where the input query is the primary determinant of difficulty. Although some benchmarks in this line include multi-turn conversations (Hu et al., 2024), routing is still decided at the level of an entire query and its response rather than at each step. They are less suited to agentic POMDPs, where the risk of using the local model changes after each action and observation. R2V differs by learning a step-level router whose input features summarize the current context, SLM uncertainty, verifier scores, and recent execution signals. This makes routing a stateful decision over the trajectory rather than a static decision over the initial prompt.

Agentic environments and sequential tool use. Modern LLM agents operate through iterative interaction with external tools, web pages, terminals, or simulated environments (Mialon et al., 2024; Yang et al., 2024a; Zhou et al., 2024b; Shridhar et al., 2021a). In such settings, the agent must recover from partial observations, invalid actions, tool failures, and compounding mistakes. Prior work primarily improves the agent policy or the agent-computer interface; R2V instead studies when a cheap local policy should defer to a stronger teacher during the trajectory. This distinction is important because the frontier model need not be used uniformly: many steps are routine, while a small number of states determine whether the episode recovers or fails. R2V formalizes this as residual-risk estimation for a fixed deployed SLM.

Process reward models and verifiers. Outcome reward models provide only trajectory-level feedback, whereas process reward models supervise intermediate reasoning or action steps (Lightman et al., 2023). This has improved mathematical reasoning (Wang et al., 2024), code generation and test-time search (Snell et al., 2025), and agentic control (Xi et al., 2025a). Execution-based verification is especially useful when available, since unit tests, terminal outputs, or environment transitions provide grounded evidence about action quality (Liu et al., 2023). R2V builds on this process-supervision view but uses the verifier in three roles: to rank SLM candidates, to construct preference pairs for recovery distillation, and to supply risk features for routing. Thus the verifier is not merely an auxiliary evaluator; it is the bridge between policy improvement and compute allocation.

Preference optimization, recovery training, and self-correction. Preference optimization methods such as DPO provide an offline alternative to reinforcement learning from scalar rewards by directly increasing the likelihood of preferred responses relative to rejected ones (Rafailov et al., 2023). Self-correction and recovery-oriented training methods similarly aim to improve behavior after an initial mistake or weak response (Kumar et al., 2025). R2V adopts the recovery perspective but changes the data-generating process: the BC policy is itself imitation-trained on a trajectory pool that already includes synthetically perturbed observations, and preference pairs are then constructed by this trained BC policy's rollouts. So training focuses on the SLM's own recoverable errors rather than generic preference data. The BC policy is frozen as the DPO reference, and the second stage optimizes only the verifier-guided DPO and consistency losses. This ordered pipeline avoids treating BC, preference learning, consistency, and routing as a single joint objective, which would entangle capability transfer with deployment-time compute allocation.

Robustness to perturbations and consistency regularization. Robust LLM training often uses perturbations to encourage invariance to prompt changes, noisy contexts, or semantically equivalent inputs (Qiang et al., 2024a; Wang et al., 2025a). In agent environments, perturbations can also represent tool flakiness, missing observations, corrupted feedback, latency-induced truncation, or distracting context. R2V uses perturbations at two levels. First, perturbed contexts are mixed into the BC training pool and then re-used during recovery distillation: this exposes failure modes that survive imitation under noise and provides states for verifier-guided DPO. Second, paired perturbation views support consistency regularization, encouraging the SLM to produce stable action distributions across different observations of the same latent state. This is complementary to the router: consistency reduces unnecessary escalation by making the SLM more stable, while the router handles the residual high-risk states that remain after distillation.

Uncertainty, calibration, and risk-sensitive control. Many routing systems rely on uncertainty proxies such as entropy, margin, or confidence. However, token-level uncertainty can be poorly aligned with downstream agent failure, especially under partial observability or when the model is confidently wrong. Calibration methods such as temperature scaling and proper scoring rules address whether predicted probabilities match empirical frequencies (Guo et al., 2017). R2V uses Brier calibration so that the router output estimates the conditional probability that teacher intervention would be useful at the current step. This calibrated probability supports a cost-sensitive threshold derived from the relative cost of SLM and LLM execution and the penalty for missed escalation.

R2V further connects model routing to risk-sensitive RL. CVaR objectives optimize the tail of the return or loss distribution rather than only its mean, and have been used to produce policies that are robust to rare but severe failures (Chow & Ghavamzadeh, 2014; Tamar et al., 2015). R2V applies this principle to hybrid SLM–LLM execution: the router is trained not only to reduce average cost, but also to avoid policies that are cheap on average while failing catastrophically under the worst

Table 14: **Comparison with representative routing, verifier, and agent-training frameworks.** R2V is distinguished by stateful step-level routing, perturbation recovery, and calibrated tail-risk control.

Method	Step-Level Routing	Stateful Agent	Process Verifier	Risk-Calib. (CVaR)	Perturb. Robustness	Recovery Training
FrugalGPT (Chen et al., 2024)	✗	✗	✗	✗	✗	✗
RouteLLM (Ong et al., 2025)	✗	✗	✗	✗	✗	✗
Hybrid LLM (Ding et al., 2024a)	✗	✗	✗	✗	✗	✗
Unified routing/cascading (Dekoninck et al., 2025)	✗	✗	✗	✗	✗	✗
EmbedLLM (Zhuang et al., 2024)	✗	✗	✗	✗	✗	✗
AgentPRM (Xi et al., 2025b)	✗	✓	✓	✗	✗	✗
SCoRe (Kumar et al., 2025)	✗	✗	✗	✗	✗	✓
R2V-Agent (ours)	✓	✓	✓	✓	✓	✓

Table 15: **Approximate wall-clock times per pipeline stage** for a single benchmark–backbone configuration on four H100 80 GB GPUs. "API" stages consume no local GPU; the router training and evaluation stages run on CPU.

Stage	Description	Hardware	Approx. time
1	Trajectory collection	API (no GPU)	3–6 h
2	Perturbation generation	CPU	$\sim$2 min
3a	Static split creation	CPU	$<$1 min
3b	Behavioral cloning (BC)	1$\times$ H100	2–3 h
4	Candidate generation (vLLM) + Verifier Scoring	1$\times$ H100	4–7 h
5	DPO preference training	4$\times$ H100	12–16 h
6	Router feature extraction	1$\times$ H100	4–7 h
7	Router training	CPU	$\sim$25 s
8	Offline evaluation	CPU	$<$1 min
9	Ablation sweeps	CPU	$<$5 min

perturbation seeds. This yields a routing interface that is both probabilistic and risk-aware, in contrast to ad hoc entropy thresholds or purely average-case cascade rules.

Positioning. Table 14 summarizes the closest points of comparison. Existing LLM routers reduce cost but are largely query-level and stateless. Agent verifiers provide useful process feedback but do not decide when to allocate frontier-model compute. Recovery-training methods improve the base policy but do not produce calibrated escalation decisions. R2V combines these strands by training a local SLM for perturbation recovery and then learning a CVaR-calibrated step router over the fixed policy's remaining failure modes.

E Compute Resources and Asset Licenses

E.1 Compute Resources

All GPU-bound stages were executed on a single server with **four NVIDIA H100 80 GB SXM GPUs**. The full pipeline was run independently for each of the three benchmarks (HumanEval+, TextWorld, TerminalBench) and each of the four SLM backbones (Gemma-9B, LLaMA-3.1-8B, Qwen2.5-7B, Qwen2.5-14B); ablation sweeps used subsets of this grid. Table 15 gives approximate wall-clock times for each pipeline stage. Stages that are purely API-driven or CPU-only are listed separately.

Total compute budget. The end-to-end pipeline (stages 1–10) requires approximately **25–40 GPU-hours per benchmark–backbone configuration**. Summing over three benchmarks and four backbones yields roughly **300–480 GPU-hours** for the main experiments reported in the paper. Ablation sweeps (CVaR grid, consistency sweep, feature ablations) add approximately 20% overhead, bringing the total to approximately **360–576 H100 GPU-hours**. The router (stage 8) and all evaluation stages (stages 9–10) run exclusively on CPU and contribute negligibly to compute cost.

Table 16: **Licenses and terms of use for all third-party assets.** API services are governed by the respective provider's terms of service (ToS). "Research use" indicates that the license or ToS permits academic non-commercial research. All assets were used in accordance with their stated licenses and intended research purposes.

Asset	Type	License / Terms	Use
Datasets and benchmarks			
EvalPlus / HumanEval+ (Liu et al., 2023)	Benchmark	Apache 2.0	Research
ALFWorld (Shridhar et al., 2021b)	Benchmark	MIT License	Research
TerminalBench (Merrill et al., 2026)	Benchmark	Apache 2.0	Research
`yoonholee/terminalbench-trajectories`	Dataset	Apache 2.0	Research
Open-weight SLM backbones			
Gemma-9B (Team, 2024)	Model	Gemma Terms of Use	Research
LLaMA-3.1-8B (Grattafiori et al., 2024)	Model	LLaMA 3.1 Community License	Research
Qwen2.5-7B (Qwen et al., 2025)	Model	Apache 2.0	Research
Qwen2.5-14B (Qwen et al., 2025)	Model	Apache 2.0	Research
API-accessed teacher models			
Gemini 3 Flash Preview	API	Google API ToS	Research
Gemini 3.1 Pro	API	Google API ToS	Research
GPT-5.5	API	OpenAI API ToS	Research
Claude Opus 4.6	API	Anthropic API ToS	Research
Key software libraries			
vLLM (Kwon et al., 2023)	Library	Apache 2.0	Research
HuggingFace Transformers	Library	Apache 2.0	Research
HuggingFace Accelerate	Library	Apache 2.0	Research
DeepSpeed	Library	Apache 2.0	Research
PyTorch	Library	BSD-style	Research

API cost. Trajectory collection (stage 1) and teacher escalation calls use the Gemini, OpenAI, and Anthropic APIs. HumanEval+ and TextWorld teacher trajectories are collected with Gemini 3 Flash Preview; TerminalBench uses a heterogeneous pool of Claude Opus 4.6, GPT-5.5, and Gemini 3.1 Pro. Estimated API expenditure for the full set of trajectories is on the order of **$200–$500** in total, dominated by TerminalBench frontier-model calls.

E.2 Asset Licenses and Terms of Use

Table 16 lists every third-party dataset, model backbone, and key software library used in R2V-Agent, along with the associated license or terms of use.