SCALING AUTOMATIC RESEARCH AGENTS VIA WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Automating empirical research is a long-standing direction of AI. Recent automatic research (AutoResearch) agents bring this goal within reach, as modern LLMs show the capability to independently implement solutions and learn from the execution outcomes. Behind these gains, post-training (especially RL) plays a central role. In this paper, we identify a fundamental tension when scaling RL for these agents: the two components of every AutoResearch trajectory (agent generation and environment execution) scale in very different manners, since all generation shares compute through batching, while each execution occupies its exclusive sandbox and real machine time. As a result, the environment execution dominates the training cost and becomes the bottleneck as trajectories grow. To resolve this tension, we propose *World Model RL* (WMRL), which replaces environment execution with a world model to remove this bottleneck. Additionally, the world model can be imperfect, as its rewards are corrupted by bias and noise. Therefore, we further equip WMRL with two mitigations, *Online Debiasing* and *Inverse-Variance Denoising*, which offset the bias and suppress the noise respectively. Theoretically, we prove that both mitigations of WMRL improve the convergence guarantee. Empirically, WMRL accelerates training by 3–4$\times$ on various tasks at different agent scales, while exceeding the performance of standard RL baselines. Moreover, our post-trained 4B and 9B agents outperform much larger open-weight agents of 48B and 120B on held-out benchmarks. Beyond AutoResearch, WMRL also transfers to post-training embodied VLA policies, demonstrating its generalizability.

1 INTRODUCTION

An Automatic Research (AutoResearch) agent is a language model that independently conducts empirical research (Lu et al., 2024; Gottweis et al., 2025; Schmidgall et al., 2025). Given a research question, it formulates an idea, implements the experiment, analyzes the outcome, and iterates (Yao et al., 2023b;a; Shinn et al., 2023). Such agents have already proven capable across various domains. In the natural sciences, for example, they design chemical syntheses and propose reaction conditions that are validated by wet-lab experiments later (Boiko et al., 2023; Bran et al., 2024; Szymanski et al., 2023); in machine learning and data science, they explore real datasets and build training pipelines that outperform human experts (Jiang et al., 2025; Guo et al., 2024; Huang et al., 2024).

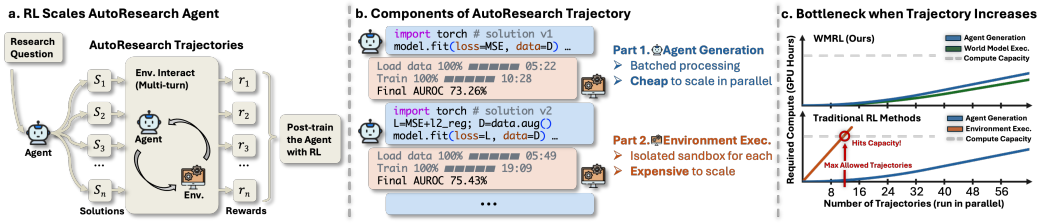

Figure 1: **Environment execution is the bottleneck in RL training of AutoResearch agents, and our WMRL removes it by replacing real execution with a fast and scalable signal from a world model.** **(a)** Per question, the agent proposes solutions S_i and receives execution rewards r_i; RL requires many trajectories. **(b)** Generation shares compute through batching, but each solution needs its own execution sandbox. **(c)** Execution saturates first in standard RL; WMRL removes this limit.

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.

In an AutoResearch task, the agent iteratively generates and executes solutions. These interactions form trajectories, and the execution outcomes provide rewards. With both trajectories and rewards in place, AutoResearch is a natural fit for reinforcement learning (RL) (Schulman et al., 2017; Shao et al., 2024; Lambert et al., 2024; Guo et al., 2025), a promising way to improve these agents.

Like most of RL's successes, training strong AutoResearch agents demands scale, i.e., massive online trajectories collected during training (Guo et al., 2025). However, we identify a fundamental tension underneath this demand: the two components of a trajectory (i.e., agent generation and environment execution) scale in different manners as the trajectory volume grows. On the generation side, rollouts are served by modern inference backends (e.g., vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024)), where batching techniques allow concurrent trajectories to share compute, making the cost of additional trajectories negligible. In contrast, execution cost cannot be amortized in this way. Every candidate solution must run in an isolated sandbox that loads the data and trains models on real GPUs (Qiang et al., 2025; Sun et al., 2026), so each additional trajectory incurs the full cost, and the total cost grows linearly with the number of trajectories. This asymmetry makes environment execution the dominant bottleneck when scaling RL for AutoResearch agents. Such a bottleneck motivates two research questions:

(i) *Can we replace the expensive environment (bottleneck) with a fast and scalable signal?*
(ii) *There is no free lunch, so what does this signal cost? (And how do we pay the bill?)*

The answer to question **(i)** is to introduce a world model (Ha & Schmidhuber, 2018; Hafner et al., 2025), which takes the environment information and the agent solution as input, and produces a simulation of real execution result as output. Such a simulation (though potentially flawed) takes only a few forward passes without real execution, so it batches and amortizes across trajectories just like generation. In RL training, we let the agent interact with the world model instead of the real environment, and receive rewards from the predicted outcomes. As a result, the expensive execution now scales as gracefully as generation and no longer bounds the scale of RL.

We answer question **(ii)** by modeling the world model's imperfection as a bias and a noise on the reward, and tracing both through the policy gradient into the convergence bound. Specifically, we consider the world model's output as an erroneous signal that deviates from real execution by a bias term b with $|b| \leq B$ and a zero-mean noise ξ with standard deviation σ. As shown in Theorem 3, the bias and the noise hinder the final convergence through two extra error terms of $O(B^2)$ and $O(\sigma^2)$ respectively. To reduce both terms, we introduce an *Online Debiasing* mechanism that recasts the world model outputs to offset the bias, and an *Inverse-Variance Denoising* mechanism to minimize the variance; we ground both mechanisms with an improved convergence guarantee (Theorem 4).

Our contributions are summarized as follows:

- We introduce the world model into the RL training of AutoResearch agents, which replaces the expensive environment execution with a few forward passes. It removes the critical scaling bottleneck and accelerates overall training by 3–4$\times$ (Section 3.2).
- We design two correction mechanisms, Online Debiasing and Inverse-Variance Denoising, to counteract the bias and the noise of the world model. With both mechanisms, the accelerated training matches or even exceeds the performance of training with the real environment (Section 3.3).
- We theoretically ground the entire framework. An imperfect world model introduces two error terms into the convergence bound (Section 4.2), and our two mechanisms provably reduce both, yielding an improved convergence guarantee (Section 4.3).
- Through experiments on various AutoResearch tasks across two agent scales, we validate both the efficiency and the performance gains (Sections 5.3 and 5.2); we further extend our method to VLA post-training tasks to demonstrate its generalizability (Section 5.4).

2 Related Work

AutoResearch agents and their post-training. Language model agents conduct chemical experiments (Boiko et al., 2023), scientific discovery (Lu et al., 2024), machine learning engineering (Jiang et al., 2025; Novikov et al., 2025; Yamada et al., 2025), data curation (Kulikov et al., 2026; Bertran et al., 2026), and recursive self-improvement (Yang et al., 2026a; Recursive, 2026). MLE-Bench, MLE-Dojo, DSBench, and MLGym evaluate this ability on real tasks (Chan et al., 2024; Qiang

et al., 2025; Jing et al., 2024; Nathani et al., 2025); software agents likewise learn through execution (Jimenez et al., 2024; Yang et al., 2024; Wang et al., 2025; Pan et al., 2025). RL objectives (Shao et al., 2024; Yu et al., 2025; Qu et al., 2026), training systems (Sheng et al., 2024), and synthetic tasks (Xie et al., 2026) improve post-training, but executing solutions for rewards remains the bottleneck. We change where rewards come from, so our method is orthogonal to these advances.

Learned reward signals and their errors. Learned signals replace costly ground truth in RLHF (Ouyang et al., 2022), LLM judges (Zheng et al., 2023), and model-based RL (Ha & Schmidhuber, 2018; Bruce et al., 2024; Hafner et al., 2025), with world models spanning physical simulation (NVIDIA, 2025), executable code (Dainese et al., 2024; Tang et al., 2024), and software environments (Sun et al., 2026). SWE-RM trains execution-free verifiers for software agents, emphasizing discrimination and calibration (Shum et al., 2025). Over-optimizing imperfect rewards can degrade true performance (Gao et al., 2023; Coste et al., 2024), and biased gradients impose a convergence floor (Ajalloeian & Stich, 2020). Prior remedies recalibrate proxies offline (Zadrozny & Elkan, 2002; Guo et al., 2017) or correct off-policy distribution mismatch (Nachum et al., 2019; Li et al., 2022). We instead use a small stream of real execution to correct reward bias and variance online, and quantify both corrections in the convergence bound.

3 METHOD

In this section, we introduce our WMRL as follows. First, we formalize the standard RL training procedure and define the notation (Section 3.1). Next, we answer the question **(i)** by incorporating the world model into the RL pipeline as a fast and scalable signal (Section 3.2). We then answer the question **(ii)** by revealing the additional error terms in the RL convergence, and then providing our mitigations (Section 3.3), which offset these error terms introduced by the world model.

3.1 PRELIMINARIES

Typically, an AutoResearch task provides the agent with the research question, data format and the output requirements. With this context, the agent policy π_θ interacts with an environment (e.g., an isolated Docker container) (Qiang et al., 2025; Jing et al., 2024) over multiple turns to iteratively refine its solution until reaching a predefined criterion. Such interactions form a trajectory τ, and the solution receives a score $r(\tau) \in [0, 1]$ that measures its quality.

To enhance the agent policy, a wide range of works leverage RL to maximize the expected score $J(\theta) := \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$ with group-relative policy optimization (GRPO) (Shao et al., 2024)[1] as follows: For each task, GRPO samples a group of $n \geq 2$ independent trajectories $\tau_1, \ldots, \tau_n$ in parallel and obtains their scores $r(\tau_i)$ by executing the final solutions. The scores are then normalized within the group into advantages $A_i := r(\tau_i) - \frac{1}{n}\sum_j r(\tau_j)$, which yield the gradient estimate as:

$$\hat{g} \;:=\; \frac{1}{n}\sum_{i=1}^{n} A_i\, s_i, \qquad s_i := \nabla_\theta \log \pi_\theta(\tau_i), \tag{1}$$

where s_i is the raw policy gradient, and the parameter is updated by $\theta_{t+1} = \theta_t + \gamma\,\hat{g}_t$ over T steps with step size γ. Throughout training, every step costs online rewards, and each of the rewards requires executing a solution in a new environment with exclusive GPU assignments.

3.2 WORLD MODEL AS AN ENVIRONMENT

To provide fast and scalable environment signals (asked in question **(i)**), we introduce a world model to replace the expensive execution, which constitutes the backbone of WMRL. In our setting, we adopt as the world model a general language model that simulates the environment (with potential errors), instantiated with the same backbone as the agent and queried for the execution outcome. Given the same task context and an agentic solution, it predicts an execution outcome yielding an estimated score $\hat{r}(\tau_i)$ in place of the true $r(\tau_i)$. Since this interface is identical to that of the real environment, the pipeline in Section 3.1 runs unchanged with the estimated scores as:

$$\hat{A}_i \;:=\; \hat{r}(\tau_i) - \frac{1}{n}\sum_j \hat{r}(\tau_j), \tag{2}$$

[1]We adopt GRPO as the most common objective for agentic RL at scale (Sheng et al., 2024; Guo et al., 2025), but our contribution primarily acts on the reward side and is orthogonal to the specific RL objective.

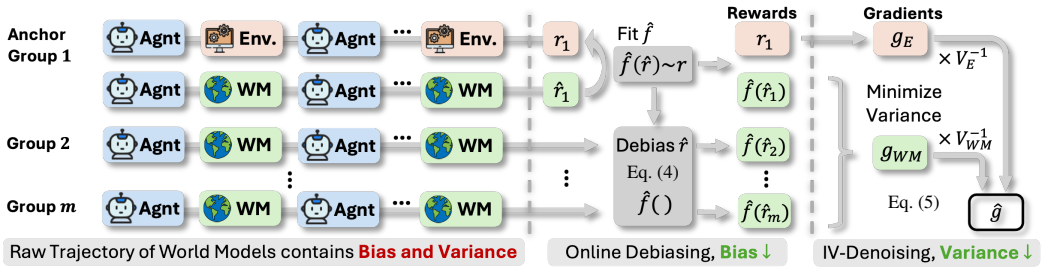

Figure 2: **WMRL corrects the world model rewards in two steps.** Each row is one of the m groups in a batch. Every group is graded by the world model into biased, noisy scores $\hat{r}$, and anchor groups are also graded by real execution. The monotone map $\hat{f}$ is fit on the score pairs (*Online Debiasing*) to remove the bias, and *Inverse-Variance Denoising* fuses g_E and g_{WM} to reduce update variance.

and the gradient estimate of Equation (1) is computed with $\hat{A}_i$ in place of A_i. In effect, the parameters now ascend the surrogate objective $\mathbb{E}_{\tau\sim\pi_\theta}[\hat{r}(\tau)]$ rather than $J(\theta)$. As a result, the execution part of a trajectory now scales as gracefully as the generation side, which directly removes the execution bottleneck. However, we note that this replacement is not free, as an imperfect predicted outcome can deviate from the real execution outcome. We characterize and mitigate such deviation in Section 3.3.

3.3 Anchor Signal for Error Correction

We now answer question **(ii)**. For any world model, the deviation of its predicted score can be decomposed into a systematic part and a random part as follows:

$$\hat{r}(\tau) \;=\; r(\tau) + b(\tau) + \xi(\tau), \tag{3}$$

where the bias $b(\tau) := \mathbb{E}[\hat{r}(\tau) \mid \tau] - r(\tau)$ collects the systematic part that survives averaging, and the noise $\xi(\tau)$ is the zero-mean remainder. Since the scores are naturally bounded, we write $B := \sup_\tau |b(\tau)|$ for the maximal magnitude of the bias and $\sigma := \sup_\tau \mathrm{std}(\xi(\tau))$ for the maximal standard deviation of the noise. The remark below summarizes how the two parts affect the convergence.

Remark 1 (The price of the world model replacement)**.** *With real execution rewards, standard RL converges with a bound $J^\star - \mathbb{E}[J(\theta_T)] \le \varepsilon(T)$ on the gap to the optimal score $J^\star := \sup_\theta J(\theta)$. With world model rewards, the same analysis gives*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \varepsilon(T) \;+\; O(B^2) \;+\; O(\sigma^2),$$

where $\varepsilon(T)$ is the standard term (Theorem 3). We next resolve the $O(B^2)$ and $O(\sigma^2)$ terms.

Our goal is to remove or reduce the two additional errors (the $O(B^2)$ and $O(\sigma^2)$ term) in Remark 1. Removing them starts with estimating them, and by Equation (3) both are defined against the true score $r(\tau)$, which the world model itself does not provide. Since the truth can only come from environment execution, we take a small step back and reintroduce a marginal amount of execution for this estimation (which we prove to be enough in Section 4). WMRL therefore keeps a thin stream of ground truth during training, which we call the *anchor signal*. A small fraction of the groups (*anchor groups*) is graded by both the world model and real execution,[2] and the resulting score pairs $\mathcal{P} = \{(\hat{r}_j, r_j)\}$ feed the two mechanisms below (Figure 2).

We first correct the **bias** by *Online Debiasing*, a monotone distribution recalibration. The score pairs reveal how world model scores drift from the truth, so we fit a mapping function to all pairs,

$$\hat{f} \;=\; \arg\min_{f} \sum_{(\hat{r}_j,\, r_j)\in\mathcal{P}} \big(f(\hat{r}_j) - r_j\big)^2, \tag{4}$$

where f ranges over monotone functions solved by isotonic regression (Zadrozny & Elkan, 2002). With this mapping, all world model scores are then recast into $\hat{f}(\hat{r}(\tau_i))$ before the advantages are formed, and $\hat{f}$ is refit as new pairs arrive on each step to track the drift over training.

For the **noise** part, since it cannot be estimated pointwise, we turn to suppress it by *Inverse-Variance Denoising*, which fuses the two reward streams to reach a lower variance. In each step the m group

[2] In practice, anchor groups make up about 10% of all groups.

indices split into the anchor set $\mathcal{G}_E$ and the world model set $\mathcal{G}_{WM}$, and every group k gives an estimate $\hat{g}_k$ of the same policy gradient, with variance V_E or V_{WM} by how it was graded. The question is therefore how to combine them, as one stream is scarce but free of world model noise and the other abundant but noisy. Motivated by Lemma 12, we use inverse-variance weighting to minimize the variance of linear fusion:

$$\hat{g} \;=\; \frac{V_E^{-1}\, g_E \;+\; V_{WM}^{-1}\, g_{WM}}{V_E^{-1}|\mathcal{G}_E| \;+\; V_{WM}^{-1}|\mathcal{G}_{WM}|}, \qquad g_E := \sum_{k\in\mathcal{G}_E} \hat{g}_k, \quad g_{WM} := \sum_{k\in\mathcal{G}_{WM}} \hat{g}_k, \tag{5}$$

where the stream gradients g_E and g_{WM} sum the group estimates $\hat{g}_k = \frac{1}{n}\sum_{i=1}^{n} A_{k,i}\, s_{k,i}$, each applying Equation (1) to group k with the true scores on $\mathcal{G}_E$ and the calibrated scores on $\mathcal{G}_{WM}$.

Equation (5) is the conceptual form of the fusion. In implementation, we approximate the unknown variance ratio $\rho := V_{WM}/V_E$ by $\hat{\rho}$, giving $\hat{g} = (\hat{\rho}\, g_E + g_{WM})/(\hat{\rho}\,|\mathcal{G}_E| + |\mathcal{G}_{WM}|)$. The analysis in Section 4 bounds the additional gradient noise by a second-moment term proportional to σ^2. This motivates the working approximation $\rho \approx 1 + c\,\sigma^2$. We form a noise proxy $\hat{\eta}^2$ from the mean squared residual between calibrated and true scores on anchor groups, and normalize it by its warmup value $\hat{\eta}^2_{\text{cal}}$ to obtain $\hat{\rho} = 1 + \hat{\eta}^2/\hat{\eta}^2_{\text{cal}}$. To assess sensitivity to this approximation, we show that the first-order term of the fused-variance increase is zero at the optimal weight (Remark 8). Small weight errors therefore incur a quadratic variance penalty, with the excess below three percent at the calibration point in our runs (Appendix D). Sections 4 and 5 analyze and test both mitigations.

3.4 Discussion

The residual-based weights also have a direct interpretation. On the anchor groups, the tracked residual $\hat{\eta}^2$ measures how far the calibrated predictions fall from the true scores, and thereby audits the world model throughout training. When the audit worsens, the anchor weight in Equation (5) rises. Consider a task whose outcome hinges on randomness that no reading of the solution reveals. In this case, $\hat{\eta}^2$ remains large and the gradient is carried by the anchor groups, so WMRL degrades toward the standard GRPO of Section 3.1 instead of learning from a corrupted signal. This fallback follows from the measurement itself rather than a mixing ratio fixed in advance.

Additionally, although the design targets the execution bottleneck of AutoResearch, it is not limited to this scenario. The construction requires only that rewards are expensive to execute yet predictable from the artifacts the agent produces, and that a small stream of ground truth stays available for anchoring. Post-training embodied policies is one such instance, where real rollouts are slow while learned simulators are cheap, and we validate this transfer on VLA tasks in Section 5.

4 Theoretical Analysis

In this section, we first analyze the additional error terms that an imperfect world model introduces, and then justify our mitigation by a convergence analysis. We first bound the convergence of training on world model rewards alone in Section 4.2 (Theorem 3), and then bound the convergence of WMRL in Section 4.3 (Theorem 4), and compare the two term by term. We put the full proofs and auxiliary lemmas in Appendix B.

4.1 Setup

We analyze the RL training procedure of Section 3.1, namely T ascent steps with step size γ on the gradient estimator of Equation (1), and we measure progress by the expected gap to the optimal score, $J^\star - \mathbb{E}[J(\theta_T)]$ with $J^\star := \sup_\theta J(\theta)$ and $\Delta_0 := J^\star - J(\theta_0)$. We use one standard assumption.

Assumption 2 (Regularity). *J is L-smooth with gradient domination $\|\nabla J(\theta)\|^2 \geq 2\mu\,(J^\star - J(\theta))$ for some $\mu > 0$, and log-likelihood gradients satisfy $\|\nabla_\theta \log \pi_\theta(\tau)\| \leq M$ for all τ.*

The world model contributes the bias magnitude B and noise standard deviation σ of Equation (3). Since the gradient estimator is linear in the rewards, we separate a bounded bias perturbation from a stochastic gradient component. The additional noise has second moment at most $4M^2\sigma^2/n$, which motivates the variance-ratio approximation of Section 3.3. In the convergence bounds, V_E and V_{WM} bound the variances of the stochastic components; the bias perturbation is controlled separately. Appendix B details the decomposition and scope of ideal fusion. We take $\gamma \leq 1/(8L)$.

4.2 The Cost of World Model Rewards

We first consider the setting of Section 3.2, where every group is graded by the world model and no real execution takes place, so the rewards carry the bias and the noise of Equation (3) at every step. Since the gradient estimator built on Equation (2) is linear in the rewards, the two deviations reach the gradient in different ways, the bias shifting its mean and the noise inflating its variance, and they therefore enter the convergence bound as the two separate terms announced in Remark 1.

Theorem 3 (Convergence with world model rewards)**.** *Under Assumption 2, GRPO trained on world model rewards satisfies*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; \underbrace{O\big(M^2 B^2\big)}_{\text{world model bias}} \;+\; \underbrace{O\big(\gamma\, V_{WM}\big)}_{\text{world model variance}} ,$$

where V_{WM} bounds the stochastic variance, with $O(\sigma^2)$ additional noise (Remark 1). The hidden constants depend only on L and μ. The proof is in Appendix B.2.

The first term is the standard geometric decay of the noise-free RL setting and vanishes as training proceeds. The bias term is independent of T and γ, so it remains in the bound even with longer training or a smaller step size. The variance term grows with the noise through V_{WM} and increases the residual in the bound. Both terms need correction.

4.3 The Effect of Our Mitigation

We now analyze WMRL, whose two corrections from Section 3.3 improve one term of Theorem 3 each. Both corrections rely on the anchor signal, where real execution grades a small fraction of the groups and yields score pairs throughout training. With these pairs, the Online Debiasing of Equation (4) keeps shrinking the residual bias, at a speed captured by a constant T_0 given in Appendix B.3. The Inverse-Variance Denoising of Equation (5) then fuses the two gradient streams to reduce gradient variance. Here, $\widetilde{O}$ hides logarithmic factors.

Theorem 4 (Convergence of WMRL)**.** *Under Assumption 2 and the conditions of Appendix B.3, WMRL with ideal variance-based fusion satisfies with high probability*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; \underbrace{\widetilde{O}\Big(\frac{M^2 B^2}{1 + T/T_0}\Big)}_{\text{contractive bias}} \;+\; \underbrace{O\Big(\frac{\gamma\, V_{WM}}{1 + V_{WM}/V_E}\Big)}_{\text{reduced variance}} .$$

The proof is in Appendix B.3.

The two bounds share the same leading term, while our corrections reduce the two additional error terms. Online Debiasing divides the bias term by $1 + T/T_0$, so its contribution shrinks as training proceeds. Inverse-Variance Denoising divides the variance term by $1 + V_{WM}/V_E$, with a larger reduction when the anchor stream has lower variance relative to the world model stream. Together, these reductions tighten the bound on the optimality gap and improve the convergence guarantee.

5 Experiments

We empirically validate whether WMRL delivers the promised speedup (question **(i)**) and preserves final performance (question **(ii)**). We compare held-out performance in Section 5.2 (Table 1), analyze training speedup in Section 5.3 (Figure 3), test VLA transfer in Section 5.4 (Table 2), and study the corrections through ablations (Section 5.5, Figure 4) and calibration analysis (Section 5.6, Figure 5).

5.1 Experimental Setup

Benchmarks. We evaluate WMRL on various domains, including AutoResearch tasks and vision-language-action (VLA) post-training tasks. In the AutoResearch evaluation, we use MLE-Dojo (Qiang et al., 2025), an interactive superset of MLE-Bench (Chan et al., 2024) built on Kaggle machine learning competitions, and follow the common practice (Liu et al., 2025; Cai et al., 2026; Zhou et al., 2026) of manually dividing the pool into MLE-Dojo (train) and MLE-Dojo (test) in a category-balanced way, since the original test split of MLE-Bench has several evaluation issues (Appendix C.1).

Table 1: **Main results (leaderboard percentile in %, higher is better).** Both benchmarks hold out tasks never trained on. Scores are per-task avg@8, averaged within each category, and **Avg** is the mean over categories. **Bold** marks the best value per column, and green (red) arrows mark the gain (drop) of WMRL over same-scale GRPO.

	MLE-Dojo (test)				DSBench				
Method	Tab	Text	Img	**Avg**	Bin-Cls	Multi-Cls	Regress	Other	**Avg**
Baseline Models									
Qwen3.5-4B	13.7	8.0	0.2	7.3	21.9	11.8	25.8	8.8	17.1
Qwen3.5-9B	17.4	8.1	3.4	9.6	29.1	16.0	36.3	14.1	23.9
Large Agents									
Kimi-48B-A3B	12.3	10.1	2.0	8.1	20.3	10.0	29.0	9.8	17.3
Nemotron-120B-A12B	35.1	18.8	**7.6**	20.5	38.7	26.6	**40.7**	20.6	31.7
RL in Real Environment									
Qwen3.5-4B-GRPO	24.8	16.6	4.3	15.2	34.3	17.8	28.3	22.4	25.7
Qwen3.5-9B-GRPO	33.3	17.1	5.9	18.8	40.9	26.8	40.4	16.7	31.2
RL with Pure World Model									
Qwen3.5-4B-WM	21.2	14.6	3.0	12.9	27.1	16.9	32.5	16.0	23.1
Qwen3.5-9B-WM	28.0	15.9	4.5	16.1	35.4	21.8	37.9	16.9	28.0
WMRL									
Qwen3.5-4B-WMRL	28.0 ↑3.2	16.1 ↓0.5	5.2 ↑0.9	16.4 ↑1.2	35.8 ↑1.5	22.3 ↑4.5	32.3 ↑4.0	**24.7** ↑2.3	28.8 ↑3.1
Qwen3.5-9B-WMRL	**38.0** ↑4.7	**19.4** ↑2.3	7.4 ↑1.5	**21.6** ↑2.8	**45.4** ↑4.5	**26.9** ↑0.1	39.2 ↓1.2	19.8 ↑3.1	**32.8** ↑1.6

We also put the selection rule and the full task lists in Appendix C.2. Beyond the MLE-style tasks, we further evaluate on DSBench (Jing et al., 2024), a data science benchmark also built on Kaggle modeling tasks and fully disjoint from our training set. Appendix C.4 explains why these tasks target our research loop and execution bottleneck. In the VLA evaluation, we use LIBERO-Long (Liu et al., 2023), a standard simulated manipulation benchmark, where the policy controls a robot arm to complete long-horizon tasks from image observations and language instructions.

Models and training. For the AutoResearch tasks, we post-train research agents at two scales, Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026), with the RL procedure of Section 3. For the world model, we prompt a frozen copy of the same pretrained backbone at each scale for outcome prediction (Appendix E.1). This controls additional knowledge transfer from an external simulator; Appendix C.3 explains the choice and compatibility with other world models. Throughout training, only the research agent is updated. All the runs use identical A100 GPU allocations, and we report the total training compute in GPU-hours. For the VLA tasks, the agent is MiniVLA-1B (Belkhale & Sadigh, 2024), which is a compact vision-language-action model pretrained on LIBERO-90, and the world model is Robometer (Liang et al., 2026), an off-the-shelf VLM that predicts task success rate from images. We first finetune the agent on the official demonstrations of each task, and then post-train it with the same RL procedure. Details and hyperparameters are in Appendix D.

Metrics. For the AutoResearch tasks, we adopt the official metric of MLE-Dojo (Qiang et al., 2025; Cai et al., 2026), which scores each submission by its percentile on the corresponding real competition leaderboard. The percentile is naturally normalized (within $0 - 1$), so scores are comparable across tasks. For the VLA tasks, we score each rollout by whether it succeeds. In both domains, task performance is averaged over eight attempts (avg@8).

Baselines. We compare WMRL with four types of baselines. The untrained bases Qwen3.5-4B and Qwen3.5-9B give the starting level of the backbone before any post-training. The large open-weight agents Kimi-48B-A3B and Nemotron-120B-A12B serve as strong off-the-shelf references and test whether model scale alone can replace post-training. GRPO in the real environment is the standard full-cost training that WMRL aims to match, and is our primary comparison. The pure world model baseline trains on predicted rewards alone and shows what the raw world model delivers without any correction. All trained models share the scaffold, the hyperparameters, and the data split.

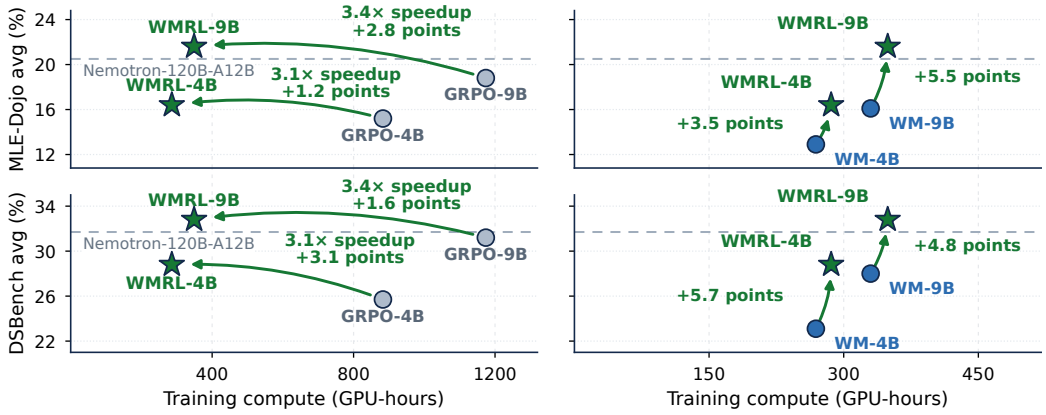

Figure 3: **Training speedup and performance on MLE-Dojo (top) and DSBench (bottom).** Left: WMRL ★ versus GRPO ●. Right: WMRL ★ versus RL with a pure world model ●. See Table 1 for details. Arrows match agent scales; dashed horizontal lines mark Nemotron-120B-A12B.

Table 2: **VLA post-training on LIBERO-Long (success rate in %).** Overall averages over seen and unseen initial states. Best@8/All@8 require at least one/all eight successes. All RL rows start from SFT; formatting follows Table 1.

	In-Domain			Out-of-Distribution			
Method	Avg	Best@8	All@8	Avg	Best@8	All@8	**Overall**
Baseline Models							
MiniVLA-1B	5.6	5.6	5.6	2.9	2.9	2.9	3.8
MiniVLA-1B-SFT	37.3	41.3	33.1	37.5	44.7	32.1	37.4
RL in Real Environment							
MiniVLA-1B-GRPO	39.3	**48.8**	34.4	37.8	45.3	31.2	38.3
RL with Pure World Model							
MiniVLA-1B-WM	39.1	45.6	34.4	39.3	45.6	34.1	39.2
WMRL							
MiniVLA-1B-WMRL	**41.2** ↑1.9	47.5 ↓1.3	**37.5** ↑3.1	**41.2** ↑3.4	**48.8** ↑3.5	**37.1** ↑5.9	**41.2** ↑2.9

5.2 Main Results on AutoResearch Tasks

Table 1 compares WMRL with the four types of baselines of Section 5.1, at both scales and on both benchmarks, and together with Figure 3 examines whether WMRL delivers the promised speedup and whether the cheap signal costs any final performance. The answer is positive on both counts. First, WMRL cuts the training compute of real-execution GRPO by $3.1\times$ and $3.4\times$ and still scores higher on every benchmark, with gains of up to 3.1 points, so the saving arrives at no cost in final performance. Second, the post-trained agents beat far larger off-the-shelf agents, as our 4B model surpasses the 48B agent and our 9B model surpasses the 120B agent on both averages.

5.3 Training Speedup

Figure 3 compares total training compute and final performance. WMRL reduces compute from 883 to 286 GPU-hours for the 4B agent and from 1174 to 349 for the 9B agent. With identical A100 GPU allocations, these reductions correspond to $3.1\times$ and $3.4\times$ training speedups. Average percentile also improves by 1.2/2.8 points on MLE-Dojo and 3.1/1.6 on DSBench (4B/9B), so the saving arrives at no cost in final performance. Compared with RL with a pure world model, WMRL gains 3.5/5.5 points on MLE-Dojo and 5.7/4.8 on DSBench with only $6.3\%/5.8\%$ more compute.

5.4 Generalization Ability on Embodied VLAs

We next test whether WMRL transfers to other domains, and we pick vision-language-action (VLA) post-training, an active line of embodied learning (Kim et al., 2024; Physical Intelligence et al., 2025; Bai et al., 2026). The two calibration mechanisms are not specific to AutoResearch, so we

apply the same recipe to the VLA setup of Section 5.1. Here Robometer predicts a progress value for eight frames sampled from each rollout as the dense reward, and the single sparse success that the environment returns at the end of a rollout serves as the anchor. The resulting success rates in Table 2 repeat the pattern of Table 1. Either signal alone barely moves the policy, as RL on the sparse outcome adds 0.9 points over the SFT baseline and RL on the raw world model signal adds 1.8. WMRL combines the two and lifts the overall success rate by 3.8 points, with the largest margin on unseen initial states. The recipe thus generalizes across domains, architectures, and reward types.

5.5 Ablation Study

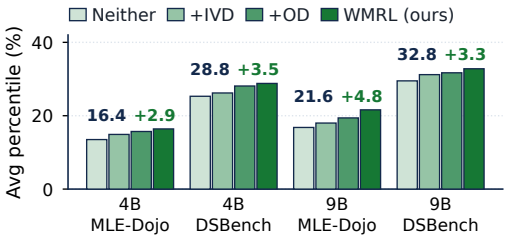

Figure 4: **Ablation on the two components.** OD: Online Debiasing; IVD: Inverse-Variance Denoising. Labels show WMRL's average percentile.

Figure 4 ablates the two corrections at both scales. All runs use the same two reward streams at the same ratio and differ only in the active corrections. Each correction helps alone, with Online Debiasing contributing the larger gain in every setting. Adding Inverse-Variance Denoising to Online Debiasing yields a further 0.7–2.2 points, with the largest gain on 9B MLE-Dojo. Combining both improves over the uncorrected mixture by 2.9–4.8 points and performs best across both scales and benchmarks, showing that the two corrections are complementary.

5.6 Effect of the Fitted Mapping

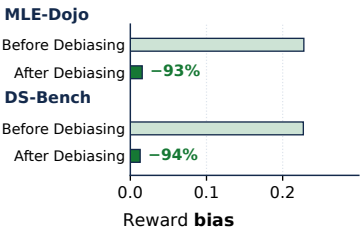

Figure 5: **Reward bias reduction.**

To examine the fitted mapping, we use anchor groups from complete trajectories of the 4B agent to fit a monotone mapping function $\hat{f}$, following Equation (4). The learned function adjusts world model scores to the scale of real rewards without updating the world model itself. On held-out groups, we compare rewards before and after calibration, $\hat{r}$ and $\hat{f}(\hat{r})$, with ground truth rewards $r \in [0, 1]$. Using separate groups tests whether the correction extends beyond the anchor samples used for fitting. Figure 5 shows reward bias reductions of 93% on MLE-Dojo and 94% on DSBench, indicating that the fitted mapping corrects much of the systematic mismatch in reward scale on both benchmarks. A reliable ground truth policy gradient is unavailable in this diagnostic, so we only focus here on reward calibration rather than error relative to the true gradient.

6 Conclusion

This work scales RL for AutoResearch agents by replacing the expensive environment execution with a world model and correcting its bias and noise through a small anchored stream of real execution. The two corrections reduce the bias and variance terms in the convergence bound. Empirically, they cut the training compute by three to four times while matching or exceeding full real-execution RL at two scales. The VLA results suggest wider applicability wherever execution limits RL scaling.

AI Use Statement

We used generative AI tools to assist with writing, figures, experimental design and implementation, result analysis, and the checking and refinement of theoretical arguments and proofs. The authors retain responsibility for all claims, results, and other content in this paper.

REFERENCES

Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of SGD with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. In *International Conference on Machine Learning*, 2026.

Suneel Belkhale and Dorsa Sadigh. MiniVLA: A better VLA with a smaller footprint. `https://github.com/Stanford-ILIAD/openvla-mini`, 2024.

Martin Bertran, Riccardo Fogliato, and Zhiwei Steven Wu. Many ai analysts, one dataset: Navigating the agentic data science multiverse. *Proceedings of the National Academy of Sciences*, 123(29): e2606495123, 2026.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning*, 2024.

Yuzhu Cai, Zexi Liu, Xinyu Zhu, Cheng Wang, Yanfeng Wang, and Siheng Chen. AceGRPO: Adaptive curriculum enhanced group relative policy optimization for autonomous machine learning engineering. *arXiv preprint arXiv:2602.07906*, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, et al. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, 2019.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, 2024.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. In *Advances in Neural Information Processing Systems*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 10835–10866, 2023.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1321–1330, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated data science by empowering large language models with case-based reasoning. In *International Conference on Machine Learning*, 2024.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *International Conference on Machine Learning*, 2024.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, et al. DSBench: How far are data science agents from becoming data science experts? *arXiv preprint arXiv:2409.07703*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, 2024.

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

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. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Jiachen Li, Shuo Cheng, Zhenyu Liao, Huayan Wang, William Yang Wang, and Qinxun Bai. Off-policy reinforcement learning with optimistic exploration and distribution correction. In *Deep Reinforcement Learning Workshop, NeurIPS*, 2022.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

Zexi Liu, Jingyi Chai, Xinyu Zhu, Shuo Tang, Rui Ye, Bo Zhang, Lei Bai, and Siheng Chen. ML-Agent: Reinforcing LLM agents for autonomous machine learning engineering. *arXiv preprint arXiv:2505.23723*, 2025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, 2019.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, et al. MLGym: A new framework and benchmark for advancing AI research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

NVIDIA. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025.

OpenAI. MLE-bench: Evaluating machine learning agents on machine learning engineering. `https://openai.com/index/mle-bench/`, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *International Conference on Machine Learning*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, et al. MLE-Dojo: Interactive environments for empowering LLM agents in machine learning engineering. *arXiv preprint arXiv:2505.07782*, 2025.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Qwen Team. Qwen3.5. `https://qwen.ai/blog?id=qwen3.5`, 2026.

Recursive. First steps toward automated AI research. `https://www.recursive.com/articles/first-steps-toward-automated-ai-research`, 2026.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, et al. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

KaShun Shum, Binyuan Hui, Jiawei Chen, Lei Zhang, X. W., Jiaxi Yang, Yuzhen Huang, Junyang Lin, and Junxian He. SWE-RM: Execution-free feedback for software engineering agents. *arXiv preprint arXiv:2512.21919*, 2025.

Shuang Sun, Huatong Song, Lisheng Huang, Jinhao Jiang, Ran Le, Zhihao Lv, Zongchao Chen, Yiwen Hu, Wenyang Luo, Wayne Xin Zhao, Yang Song, Hongteng Xu, Tao Zhang, and Ji-Rong Wen. SWE-World: Building software engineering agents in docker-free environments. *arXiv preprint arXiv:2602.03419*, 2026.

Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of novel inorganic materials. *Nature*, 624:86–91, 2023.

Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. In *Advances in Neural Information Processing Systems*, 2024.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025.

Jian Xie, Tianhe Lin, Zilu Wang, Yuting Ning, Yuekun Yao, Tianci Xue, Zhehao Zhang, Zhongyang Li, Kai Zhang, Yufan Wu, Shijie Chen, Boyu Gou, Mingzhe Han, Yifei Wang, Vint Lee, Xinpeng Wei, Xiangjun Wang, Yu Su, and Huan Sun. QUEST: Training frontier deep research agents with fully synthetic tasks. *arXiv preprint arXiv:2605.24218*, 2026.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

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*, 2024.

Junlin Yang, Che Jiang, Yu Fu, Tianwei Luo, Can Ren, Weizhi Wang, Kaikai Zhao, Hongyi Liu, et al. Frontis-MA1: Training an AI4AI model towards recursive self-improvement in machine learning engineering. *arXiv preprint arXiv:2607.28568*, 2026a.

Zherui Yang, Fan Liu, and Hao Liu. DSWorld: A data science world model for efficient autonomous agents. *arXiv preprint arXiv:2607.15901*, 2026b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

Cun-Hui Zhang. Risk bounds in isotonic regression. *The Annals of Statistics*, 30(2):528–555, 2002.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Datasets and Benchmarks Track*, volume 36, pp. 46595–46623, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Jiayi Liu, Xiangjun Fan, Zhuokai Zhao, and Hong Yan. Synthetic sandbox for training machine learning engineering agents. *arXiv preprint arXiv:2604.04872*, 2026.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, et al. Qwen-AgentWorld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026.

APPENDIX

A Notation **16**

B Proofs **16**

B.1 Warm-up: convergence of standard RL 16

B.2 Proof of Theorem 3 17

B.3 Proof of Theorem 4 and the comparison 18

B.4 Auxiliary lemmas 19

C Justification of Experimental Choices **23**

C.1 Known Benchmark Limitations 23

C.2 Data Splits and Full Task Lists 23

C.3 Choice of World Models 25

C.4 Choice of Evaluation Tasks 26

D Experimental Details **26**

E Prompts **27**

E.1 AutoResearch Agent 27

E.2 VLA Agent 31

A NOTATION

Table 3 collects the notation of the main text, in order of first appearance.

Table 3: **Notation used in the main text.**

Symbol	Meaning
τ, $r(\tau)$	a trajectory and its true score from real execution, $r(\tau) \in [0, 1]$
π_θ, $J(\theta)$	the agent policy and its expected score $\mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$
n, A_i, s_i	group size, the advantage, and the policy gradient of trajectory τ_i
$\hat{g}$, γ, T	the gradient estimate, the step size, and the number of training steps
$\hat{r}(\tau)$, $\hat{A}_i$	the world model score and the advantage computed from it
$b(\tau)$, $\xi(\tau)$	the bias and the zero-mean noise of the world model score
B, σ	the maximal bias magnitude and the maximal noise standard deviation
$\mathcal{P}$	the pool of score pairs $(\hat{r}_j, r_j)$ collected on anchor groups
$\hat{f}$	the monotone recalibration map fit on $\mathcal{P}$
m, $\mathcal{G}_E$, $\mathcal{G}_{WM}$	the groups per step and the index sets of the anchor and world model ones
g_E, g_{WM}	the sums of the group estimates $\hat{g}_k$ over $\mathcal{G}_E$ and over $\mathcal{G}_{WM}$
V_E, V_{WM}	per-group variance scales for fusion; stochastic-component variance bounds in the convergence analysis
ρ, $\hat{\rho}$	the ideal variance ratio V_{WM}/V_E and its residual-based approximation
c	the scale in the working approximation $\rho \approx 1 + c\sigma^2$
$\hat{\eta}^2$, $\hat{\eta}^2_{\text{cal}}$	the tracked residual of the calibrated scores and its warmup value
$J^\star$, Δ_0	the optimal score and the initial gap $J^\star - J(\theta_0)$
L, μ, M	the smoothness, gradient domination, and gradient bound constants
$\varepsilon(T)$	the convergence term of standard RL (Proposition 5)
T_0	the recalibration time scale of Theorem 4

B PROOFS

This appendix proves every claim of Section 4, in the order of the story. Appendix B.1 derives the standard convergence term $\varepsilon(T)$ of Remark 1 for training with real execution. Appendix B.2 proves Theorem 3 for training on world model rewards, and Appendix B.3 proves Theorem 4 for WMRL and concludes with the term-by-term comparison. The auxiliary lemmas invoked along the way are only cited there, and their statements with full proofs are collected in Appendix B.4. Throughout, the variance of a random vector X is the scalar $\mathrm{Var}(X) := \mathbb{E}\|X - \mathbb{E}X\|^2$, the trace of its covariance matrix, which satisfies $\mathbb{E}\|X\|^2 = \|\mathbb{E}X\|^2 + \mathrm{Var}(X)$.

Scope of the variance analysis. The analysis decomposes a world model gradient into a stochastic component and a bounded bias perturbation. Its variance parameter controls the former; it is not an identity for the variance of the full biased gradient. In particular, a perturbation that is deterministic given the trajectories can still vary across trajectories. Lemma 10 retains this distinction, while the perturbation term in Lemma 9 controls its effect on convergence.

Theorem 4 analyzes ideal variance-based fusion. For each update, the policy, calibration map and weights are fixed before drawing independent groups; this can be formalized by conditioning on the preceding fitting data. Under this convention, V_E and V_{WM} are fixed positive bounds on the per-group stochastic variances. They may be chosen uniformly over the updates under consideration. Using the same groups both to fit the map or weights and to form the fused estimate can introduce dependence not covered by this calculation. The practical residual rule motivates an approximation to the ideal weights; its sensitivity is analyzed separately in Remark 8.

B.1 WARM-UP: CONVERGENCE OF STANDARD RL

We first derive the bound that training with real execution satisfies. It contains no contribution from the world model and is exactly the $\varepsilon(T)$ of Remark 1.

Proposition 5 (Standard RL)**.** *Under Assumption 2 and $\gamma \le 1/(8L)$, GRPO trained on real execution rewards satisfies*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; \frac{4L\gamma}{\mu}\, V_E \;=:\; \varepsilon(T).$$

Proof. We proceed in three steps.

Step 1, the estimate is clean. Let $\mathbb{E}_t$ denote expectation conditioned on θ_t, so that $J(\theta_t)$ and its gradient are fixed under $\mathbb{E}_t$. By Lemma 10 (i), proved in Appendix B.4, the estimator of Equation (1) satisfies $\mathbb{E}_t[\hat{g}_t] = \kappa \nabla J(\theta_t)$ with $\kappa = 1 - \frac{1}{n} \in [\frac{1}{2}, 1)$ and $\mathrm{Var}(\hat{g}_t \mid \theta_t) \le V_E$. Since $\|\mathbb{E}_t \hat{g}_t\| = \kappa \|\nabla J(\theta_t)\| \le \|\nabla J(\theta_t)\|$, the second moment obeys $\mathbb{E}_t\|\hat{g}_t\|^2 = \|\mathbb{E}_t \hat{g}_t\|^2 + \mathrm{Var}(\hat{g}_t \mid \theta_t) \le \|\nabla J(\theta_t)\|^2 + V_E$.

Step 2, one-step progress. Write $\nabla := \nabla J(\theta_t)$. By L-smoothness of J,

$$J(\theta_{t+1}) \;\ge\; J(\theta_t) + \gamma \langle \nabla, \hat{g}_t \rangle - \frac{L\gamma^2}{2}\, \|\hat{g}_t\|^2.$$

Taking expectations conditioned on θ_t and inserting Step 1,

$$\mathbb{E}_t\, J(\theta_{t+1}) \;\ge\; J(\theta_t) + \gamma\Big(\kappa - \tfrac{L\gamma}{2}\Big)\|\nabla\|^2 - \frac{L\gamma^2}{2}\, V_E \;\ge\; J(\theta_t) + \frac{\gamma}{8}\, \|\nabla\|^2 - \frac{L\gamma^2}{2}\, V_E,$$

where the last step uses $\kappa \ge \frac{1}{2}$ and $L\gamma \le \frac{1}{8}$.

Step 3, gradient domination and unrolling. By Assumption 2, $\|\nabla\|^2 \ge 2\mu\,(J^\star - J(\theta_t))$. Writing $\delta_t := J^\star - \mathbb{E}[J(\theta_t)]$ and taking total expectations,

$$\delta_{t+1} \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)\, \delta_t + \frac{L\gamma^2}{2}\, V_E.$$

Unrolling over $t = 0, \dots, T-1$ and bounding the geometric sum by $\sum_{k \ge 0}(1 - \frac{\gamma\mu}{4})^k = \frac{4}{\gamma\mu}$ gives $\delta_T \le (1 - \frac{\gamma\mu}{4})^T \Delta_0 + \frac{2L\gamma}{\mu} V_E$, and we state the looser constant $\frac{4L\gamma}{\mu}$ to share constants with the perturbed case below. □

B.2 Proof of Theorem 3

With predicted rewards the estimate is no longer clean. By Lemma 10, stated and proved in Appendix B.4, the estimator computed with predicted scores decomposes as $\hat{g}_k^{WM} = \hat{g}_k + g_b + g_\xi$, where g_b is a bounded trajectory-dependent perturbation induced by the bias and g_ξ is a zero-mean fluctuation induced by the noise. The recursion of Proposition 5 then goes through with two modifications, the variance grows and a perturbation term appears, which is the content of Lemma 9 in Appendix B.4.

Theorem (Convergence with world model rewards, Theorem 3 of Section 4.2 with explicit constants)**.** *Under Assumption 2, GRPO trained on world model rewards with $\gamma \le 1/(8L)$ satisfies*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; \frac{32M^2}{\mu}\, B^2 \;+\; \frac{4L\gamma}{\mu}\, V_{WM}.$$

Proof. The plan is to verify the hypotheses of Lemma 9 for the decomposition supplied by Lemma 10, and then read off the bound. By Lemma 10, the estimate computed with predicted scores splits as $u + d$ with $u := \hat{g}_k + g_\xi$ and $d := g_b$. The part u is the clean estimate plus a conditionally zero-mean fluctuation, so its mean is the unchanged $(1 - \frac{1}{n})\nabla J$, and since $\hat{g}_k$ and g_ξ are uncorrelated, $\mathrm{Var}(u) = \mathrm{Var}(\hat{g}_k) + \mathrm{Var}(g_\xi) \le V_E + 4M^2\sigma^2/n$. We may therefore choose $V_{WM} := V_E + 4M^2\sigma^2/n$ as a variance bound for u. This choice is not an equality for the variance of the full world model gradient. The part d obeys $\|d\| \le 2MB$ almost surely, which serves as the constant perturbation bound D. Applying Lemma 9 with $\kappa_t = 1 - \frac{1}{n} \ge \frac{1}{2}$, $V = V_{WM}$ and $D = 2MB$ then yields the three terms of the claim, the geometric decay, the constant bias term $\frac{8}{\mu}(2MB)^2 = \frac{32M^2}{\mu}B^2$, and the variance term $\frac{4L\gamma}{\mu} V_{WM}$. □

B.3 Proof of Theorem 4 and the comparison

Compared with Appendix B.2, two things improve for WMRL, the recalibration makes the perturbation bound shrink over time, and the fusion makes the variance harmonic. Analyzing the recalibration needs one modeling assumption beyond Assumption 2.

Assumption 6 (Score scale)**.** *The bias acts on the score scale as a monotone distortion, that is, $b(\tau) = \phi(r(\tau)) - r(\tau)$ for a non-decreasing ϕ, and the noise ξ is independent across trajectories with $|\xi| \le 1$.*

This is what makes the bias learnable by a monotone fit. How fast it is learned is quantified by Lemma 11, whose constant c_f measures how quickly the recalibration error decays. The constant T_0 of Theorem 4 is $T_0 := c_f^2/B^2$, the number of steps after which the residual bias of the recalibrated scores falls below the raw bias of the world model.

Theorem (Convergence of WMRL, Theorem 4 of Section 4.3 with explicit constants)**.** *Under Assumptions 2 and 6, the ideal fusion scope of Appendix B, and $\gamma \le 1/(8L)$, WMRL satisfies with probability at least $1-\delta$*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; \frac{64\, c_f^2 M^2}{\mu} \cdot \frac{\log(2KT/\delta)}{T} \\ +\; 4\gamma T M^2 B^2 \Big(1 - \tfrac{\gamma\mu}{4}\Big)^{T/2} \;+\; \frac{4L\gamma}{\mu}\Big(\frac{1}{V_E} + \frac{1}{V_{WM}}\Big)^{-1},$$

where V_{WM} is formed with the post-recalibration noise and c_f is the constant of Lemma 11. The third term decays exponentially in T, and the whole perturbation is at most $\frac{8}{\mu}M^2B^2$ at every T. Since $\min(B^2, c_f^2/T)$ and $B^2/(1+T/T_0)$ with $T_0 := c_f^2/B^2$ agree up to a factor of two, this is the form stated in Section 4.3.

Proof. The proof has two parts. We first verify that the fused estimate satisfies the hypotheses of Lemma 9, now with a smaller variance and a perturbation that shrinks over time, and we then bound the resulting perturbation sum, which is where the contraction of the bias comes from.

Part 1, the hypotheses. Work on the success event of Lemma 11, which has probability at least $1-\delta$. On this event the recalibrated scores have systematic error at most $\min\big(B,\, c_f\sqrt{\log(2KT/\delta)/t}\big)$ at step t, since recalibration maps into the score range, so the error never exceeds B, and the sharper bound of Lemma 11 applies once enough anchor pairs have accumulated. Applying Lemma 10 to the recalibrated scores decomposes the world model estimate as $u_{WM} + d_{WM}$ with $\mathrm{Var}(u_{WM}) \le V_{WM}$ and $\|d_{WM}\| \le 2M\min(B, c_f\sqrt{\log(2KT/\delta)/t})$, while the anchor estimate has the same mean $(1-\frac{1}{n})\nabla J$ and variance at most V_E. Under the fixed-map, independent-group scope of Appendix B, let $m_E = |\mathcal{G}_E| \ge 1$ and $m_{WM} = |\mathcal{G}_{WM}| \ge 1$. Equation (5) then gives the stochastic variance bound

$$V_{\mathrm{fuse}} \le \left(\frac{m_E}{V_E} + \frac{m_{WM}}{V_{WM}}\right)^{-1} \le \left(\frac{1}{V_E} + \frac{1}{V_{WM}}\right)^{-1}.$$

The first inequality follows by adding independent group variances with their squared weights; it is an equality when the variance bounds are attained. The second yields the simpler bound displayed in the theorem. Since the normalized weights sum to one, fusion preserves the common mean and does not enlarge the bound on d_{WM}. Lemma 9 therefore applies with $\kappa_t = 1 - \frac{1}{n}$, $V = (1/V_E + 1/V_{WM})^{-1}$ and $D_t = 2M\min(B, c_f\sqrt{\log(2KT/\delta)/t})$. This comparison holds for the given group counts; it does not assert superiority to reallocating the same total budget entirely to either stream.

Part 2, the perturbation sum. What remains is the middle term of Lemma 9,

$$2\gamma \sum_{t<T} \Big(1 - \tfrac{\gamma\mu}{4}\Big)^{T-1-t} D_t^2, \qquad D_t^2 \;\le\; 4M^2 \min\Big(B^2,\; c_f^2 \tfrac{\log(2KT/\delta)}{t}\Big).$$

The two factors decay in opposite directions, the geometric weight is small for early t and the summand D_t^2 is small for late t, so we split the sum at $T/2$ and use the stronger effect on each half. For $t \ge T/2$ the summand is small, $D_t^2 \le 8M^2 c_f^2 \log(2KT/\delta)/T$, and the geometric weights sum to at most $\frac{4}{\gamma\mu}$, so this half contributes at most $\frac{64 c_f^2 M^2}{\mu}\log(2KT/\delta)/T$, the contracting bias term. For $t < T/2$ the weight is small, $(1-\frac{\gamma\mu}{4})^{T-1-t} \le (1-\frac{\gamma\mu}{4})^{T/2}$, and with the crude bound

$D_t^2 \le 4M^2B^2$ over at most $T/2$ terms this half contributes at most $4\gamma TM^2B^2(1-\frac{\gamma\mu}{4})^{T/2}$, which vanishes exponentially in T. Finally, bounding the same sum with the constant $D \equiv 2MB$ instead gives $\frac{8}{\mu}M^2B^2$, so the perturbation never exceeds the floor of Theorem 3 and decays as $\widetilde{O}(M^2c_f^2/T)$, which is the contracting form stated in Section 4.3. □

Corollary 7. *Assume in addition that the recalibration map is non-expansive, so that the post-recalibration noise satisfies $\eta^2 \le \sigma^2$. Then, comparing Theorem 4 with Theorem 3 term by term, the leading terms coincide, there exists a finite T_1 such that for all $T \ge T_1$ the bias term of WMRL is strictly smaller than the floor $\frac{32M^2}{\mu}B^2$ and converges to zero as $T \to \infty$, and the variance term of WMRL is smaller by the factor $1 + V_{WM}/V_E > 1$.*

Proof. The leading terms are identical by construction. For the bias, the term of Theorem 4 falls below $\frac{32M^2}{\mu}B^2$ as soon as $\log(2KT/\delta)/T \le B^2/(2c_f^2)$ and the exponentially decaying remainder is below the same threshold, both of which hold for all T beyond some finite T_1, and the term itself converges to zero. For the variance, Lemma 12 writes the fused variance as $V_{WM}/(1 + V_{WM}/V_E)$, and non-expansiveness gives $\eta^2 \le \sigma^2$, so the V_{WM} of Theorem 4, formed with the post-recalibration noise η, is no larger than the V_{WM} of Theorem 3, formed with σ. □

Remark 8 (Plug-in weights)**.** *The residual-based ratio approximates the ideal fusion weights. We quantify the resulting variance change by expanding around the optimal weight, where the first-order term is exactly zero. Fix the calibration map and use independent groups as in Appendix B. Let m_E, m_{WM} be the group counts and $A := V_E/m_E$, $B_v := V_{WM}/m_{WM}$ the variance bounds of the two stream means. With normalized anchor weight α, their fused stochastic variance is bounded by $Q(\alpha) := A\alpha^2 + B_v(1-\alpha)^2$. Lemma 12 gives $\alpha^\star = B_v/(A+B_v)$, and direct differentiation yields*

$$Q'(\alpha^\star) = 2A\alpha^\star - 2B_v(1-\alpha^\star) = 0,$$
$$Q(\hat\alpha) - Q(\alpha^\star) = (A+B_v)(\hat\alpha - \alpha^\star)^2.$$

Thus, the first-order term vanishes because the expansion is taken at the minimum of the fusion objective. It does not require the estimated weight to be exact or unbiased.

For the ratio used in Section 3.3, write $\rho := V_{WM}/V_E$, $\varepsilon_\rho := (\hat\rho - \rho)/\rho$ and $R := m_E\rho/m_{WM}$. Since $\hat\alpha = m_E\hat\rho/(m_E\hat\rho + m_{WM})$, substitution gives

$$\frac{Q(\hat\alpha)}{Q(\alpha^\star)} = 1 + \frac{R\varepsilon_\rho^2}{\big(1+R(1+\varepsilon_\rho)\big)^2}$$
$$= 1 + \frac{R}{(1+R)^2}\varepsilon_\rho^2 + O(\varepsilon_\rho^3), \qquad \frac{R}{(1+R)^2} \le \frac{1}{4}.$$

This makes the zero first-order coefficient explicit for errors in the variance ratio as well. Small ratio errors therefore incur a quadratic excess, while the exact expression applies to any positive $\hat\rho$. If V_E, V_{WM} are exact variances, Q is the actual stochastic variance; otherwise the result concerns its quadratic upper bound. In the convergence calculation, $Q(\hat\alpha)$ replaces the ideal variance term, adding the corresponding quadratic penalty. The scales measured in our runs (Appendix D) put this excess below three percent at the calibration point.

Scope. *This sensitivity result does not establish that the reward-residual proxy is close to the ideal variance ratio in every run. Such a claim requires a separate estimate of its error; warmup normalization alone does not provide one. The result concerns stochastic variance for fixed or independently fitted weights, not total error relative to the true policy gradient.*

B.4 Auxiliary lemmas

This subsection collects the four lemmas used above. Before each statement we recall why the lemma is needed and where it is consumed.

The first lemma is the engine behind both theorems. It answers how far gradient ascent can still progress when the estimate carries a bounded perturbation and extra variance, and every convergence statement of this paper is an instance of it, Proposition 5 with $D_t \equiv 0$, Theorem 3 with $(V, D) = (V_{WM},\ 2MB)$, and Theorem 4 with the harmonic variance and a shrinking D_t. The middle term of its bound is the weighted telescoping sum that the two theorems then control in different ways.

Lemma 9 (Perturbed ascent under gradient domination)**.** *Let $\theta_{t+1} = \theta_t + \gamma \hat{g}_t$ with $\gamma \le 1/(8L)$. Suppose that, conditioned on θ_t, the estimate decomposes as $\hat{g}_t = u_t + d_t$, where $\mathbb{E}[u_t \mid \theta_t] = \kappa_t \nabla J(\theta_t)$ with $\kappa_t \in [\frac{1}{2}, 1]$, $\mathrm{Var}(u_t \mid \theta_t) \le V$, and $\|d_t\| \le D_t$ almost surely. Then*

$$J^\star - \mathbb{E}[J(\theta_T)] \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)^T \Delta_0 \;+\; 2\gamma \sum_{t=0}^{T-1} \Big(1 - \tfrac{\gamma\mu}{4}\Big)^{T-1-t} D_t^2 \;+\; \frac{4L\gamma}{\mu}\, V.$$

In particular, if $D_t \equiv D$, the middle term is at most $\frac{8}{\mu} D^2$.

Proof. The plan mirrors the three steps of Proposition 5. We lower-bound the useful progress $\langle \nabla, \hat{g}_t \rangle$, upper-bound the harmful second moment $\|\hat{g}_t\|^2$, and combine the two through smoothness into a recursion for δ_t. Write $\nabla := \nabla J(\theta_t)$ and let $\mathbb{E}_t$ denote expectation conditioned on θ_t; under $\mathbb{E}_t$ the quantities $J(\theta_t)$ and ∇ are fixed, so only the terms containing $\hat{g}_t$ are affected. By L-smoothness, $J(\theta_{t+1}) \ge J(\theta_t) + \gamma \langle \nabla, \hat{g}_t \rangle - \frac{L\gamma^2}{2} \|\hat{g}_t\|^2$.

Step 1, the progress term. Splitting $\hat{g}_t = u_t + d_t$, the inner product separates into signal and contamination, $\mathbb{E}_t \langle \nabla, \hat{g}_t \rangle = \kappa_t \|\nabla\|^2 + \langle \nabla, \mathbb{E}_t d_t \rangle$. The contamination can only be bounded by Cauchy–Schwarz, $\langle \nabla, \mathbb{E}_t d_t \rangle \ge -\|\nabla\| D_t$, and a product of two different quantities is inconvenient in a recursion, so we separate it with Young's inequality $ab \le a^2/4 + b^2$,

$$\mathbb{E}_t \langle \nabla, \hat{g}_t \rangle \ge \kappa_t \|\nabla\|^2 - \|\nabla\| D_t \ge \big(\kappa_t - \tfrac{1}{4}\big) \|\nabla\|^2 - D_t^2.$$

The choice of $\frac{1}{4}$ sacrifices a quarter of the signal to turn the cross term into a pure D_t^2.

Step 2, the second moment. We control mean and fluctuation separately. The mean obeys $\|\mathbb{E}_t \hat{g}_t\| \le \kappa_t \|\nabla\| + D_t$, so $(a+b)^2 \le 2a^2 + 2b^2$ gives $\|\mathbb{E}_t \hat{g}_t\|^2 \le 2\|\nabla\|^2 + 2D_t^2$, and the fluctuation obeys $\mathrm{Var}(\hat{g}_t) \le 2\,\mathrm{Var}(u_t) + 2\,\mathrm{Var}(d_t) \le 2V + 2D_t^2$. Adding the two, $\mathbb{E}_t \|\hat{g}_t\|^2 \le 2\|\nabla\|^2 + 4D_t^2 + 2V$.

Step 3, combine and unroll. Substituting the two bounds into the smoothness inequality, and simplifying with $\kappa_t \ge \frac{1}{2}$ and $L\gamma \le \frac{1}{8}$,

$$\begin{aligned} \mathbb{E}_t J(\theta_{t+1}) &\ge J(\theta_t) + \gamma\big(\kappa_t - \tfrac{1}{4} - L\gamma\big)\|\nabla\|^2 - \gamma D_t^2 (1 + 2L\gamma) - L\gamma^2 V \\ &\ge J(\theta_t) + \tfrac{\gamma}{8}\|\nabla\|^2 - 2\gamma D_t^2 - L\gamma^2 V. \end{aligned}$$

Gradient domination replaces the gradient by the gap, $\|\nabla\|^2 \ge 2\mu\,(J^\star - J(\theta_t))$, and taking total expectations through the tower property turns the display into a recursion on the deterministic sequence $\delta_t := J^\star - \mathbb{E}[J(\theta_t)]$,

$$\delta_{t+1} \;\le\; \Big(1 - \tfrac{\gamma\mu}{4}\Big)\, \delta_t + 2\gamma D_t^2 + L\gamma^2 V.$$

Multiplying the step-t inequality by $(1 - \frac{\gamma\mu}{4})^{T-1-t}$ and summing over $t = 0, \dots, T-1$, the intermediate δ_t cancel telescopically and the claim follows. For constant D the geometric sum $\sum_{k \ge 0} (1 - \frac{\gamma\mu}{4})^k = \frac{4}{\gamma\mu}$ bounds the middle term by $\frac{8}{\mu} D^2$ and the last term by $\frac{4L\gamma}{\mu} V$. □

The second lemma is what allows Lemma 9 to be fed with world model rewards. It converts the reward-level error model of Equation (3) into exactly the three quantities that Lemma 9 consumes, the mean of the clean part, its variance, and the almost-sure size of the perturbation, and it also bounds the noise second moment that motivates the variance-ratio approximation in Section 3.3.

Lemma 10 (From reward error to gradient error)**.** *Let $\hat{g}_k$ be computed with true scores and $\hat{g}_k^{WM}$ with predicted scores. Under Assumptions 2 and 6, $\hat{g}_k^{WM} = \hat{g}_k + g_b + g_\xi$, where*

- *(i) $\mathbb{E}[\hat{g}_k] = \big(1 - \frac{1}{n}\big) \nabla J(\theta)$ and $V_E = \mathrm{Var}(\hat{g}_k) \le M^2$;*
- *(ii) $g_b = \frac{1}{n} \sum_i (b_i - \bar{b})\, s_i$ satisfies $\|g_b\| \le 2MB$ almost surely, and vanishes whenever b is constant within the group;*
- *(iii) g_ξ has zero conditional mean and is uncorrelated with both $\hat{g}_k$ and g_b. It satisfies $\mathbb{E}\|g_\xi\|^2 \le 4M^2\sigma^2/n$, hence $\mathrm{Var}(\hat{g}_k + g_\xi) \le V_E + 4M^2\sigma^2/n$. The full estimator instead obeys $\mathrm{Var}(\hat{g}_k^{WM}) = \mathrm{Var}(\hat{g}_k + g_b) + \mathrm{Var}(g_\xi)$.*

Proof. Recall from Equation (5) that $\hat{g}_k = \frac{1}{n} \sum_i A_i s_i$, where the advantage $A_i = r_i - \frac{1}{n} \sum_j r_j$ is a linear function of the scores of its group. Abbreviate $r_i := r(\tau_i)$, $b_i := b(\tau_i)$, and $\xi_i := \xi(\tau_i)$.

The decomposition is pure algebra. Replacing every true score by the predicted one substitutes into the same linear formula, so with $\hat{r}_i = r_i + b_i + \xi_i$ the new advantage is

$$\hat{A}_i \;=\; \hat{r}_i - \frac{1}{n}\sum_j \hat{r}_j \;=\; A_i + (b_i - \bar{b}) + (\xi_i - \bar{\xi}),$$

with $\bar{b}, \bar{\xi}$ the group means. Multiplying by s_i/n and summing over i gives $\hat{g}_k^{WM} = \hat{g}_k + g_b + g_\xi$ with $g_b := \frac{1}{n}\sum_i (b_i - \bar{b}) s_i$ and $g_\xi := \frac{1}{n}\sum_i (\xi_i - \bar{\xi})\, s_i$; no probabilistic argument is involved, the three parts simply collect the score, bias, and noise contributions. It remains to control each part in the form that Lemma 9 consumes, the mean and variance of the clean part, the almost-sure size of the bias part, and the second moment of the noise part.

(i) *The clean part.* Expanding the advantage, $\mathbb{E}[A_i s_i] = \mathbb{E}[r_i s_i] - \frac{1}{n}\sum_j \mathbb{E}[r_j s_i]$. For $j \neq i$ the two trajectories are independent given the task, so the expectation factorizes, and $\mathbb{E}[s_i] = \mathbb{E}[\nabla \log \pi_\theta] = 0$ kills the term; the $j = i$ term contributes $\frac{1}{n}\mathbb{E}[r_i s_i]$. Summing over i, $\mathbb{E}[\hat{g}_k] = (1 - \frac{1}{n})\,\mathbb{E}[r(\tau)\nabla \log \pi_\theta(\tau)] = (1 - \frac{1}{n})\nabla J(\theta)$, where the last step is the policy gradient identity. For the variance, $r_i \in [0, 1]$ forces $|A_i| \leq 1$, hence $\|\hat{g}_k\| \leq \frac{1}{n}\sum_i \|s_i\| \leq M$, and a vector bounded by M has variance at most $\mathbb{E}\|\hat{g}_k\|^2 \leq M^2$, so $V_E \leq M^2$.

(ii) *The bias part.* It is deterministic given the trajectories, so what Lemma 9 needs is an almost-sure bound. Since $|b_i| \leq B$, the centered value obeys $|b_i - \bar{b}| \leq 2B$, and therefore $\|g_b\| \leq \frac{1}{n}\sum_i |b_i - \bar{b}|\, \|s_i\| \leq 2MB$. If b is constant within the group, the centering removes it entirely, $b_i - \bar{b} = 0$, which is why only the within-group variation of the bias ever reaches the gradient.

(iii) *The noise part.* It should act as extra variance, so we verify that it is conditionally zero-mean and compute its second moment. Rewrite $g_\xi = \frac{1}{n}\sum_i \xi_i (s_i - \bar{s})$ with $\bar{s} := \frac{1}{n}\sum_j s_j$. Conditioned on the trajectories the ξ_i are independent and zero-mean, so $\mathbb{E}[g_\xi \mid \tau_{1:n}] = 0$ and all cross terms of $\mathbb{E}\|g_\xi\|^2$ vanish, leaving

$$\mathbb{E}\|g_\xi\|^2 = \frac{1}{n^2}\sum_i \mathbb{E}\big[\mathbb{E}[\xi_i^2 \mid \tau]\, \|s_i - \bar{s}\|^2\big] \leq \frac{\sigma^2}{n^2} \cdot n \cdot (2M)^2 = \frac{4M^2\sigma^2}{n}.$$

The same conditional zero mean makes g_ξ uncorrelated with every function of the trajectories, including $\hat{g}_k$ and g_b. Thus $\mathrm{Var}(\hat{g}_k + g_\xi) = V_E + \mathrm{Var}(g_\xi) \leq V_E + 4M^2\sigma^2/n$. For the full world model estimate, the corresponding identity is $\mathrm{Var}(\hat{g}_k^{WM}) = \mathrm{Var}(\hat{g}_k + g_b) + \mathrm{Var}(g_\xi)$. The term g_b can vary across trajectories, so its variance and covariance with $\hat{g}_k$ cannot generally be discarded. Our convergence argument instead bounds g_b as a perturbation and uses the variance bound only for $\hat{g}_k + g_\xi$. □

The third lemma supplies the shrinking perturbation bound D_t used in the proof of Theorem 4. It quantifies how fast the isotonic recalibration of Equation (4) learns the distortion ϕ from the accumulating anchor pairs, and its $1/\sqrt{t}$ rate is what ultimately becomes the contraction of the bias term.

Lemma 11 (Recalibration error)**.** *Under Assumption 6, with probability at least $1 - \delta$ simultaneously for all $1 \leq t \leq T$, the recalibration map $\hat{f}_t$ is well defined once enough anchor pairs have accumulated, and the systematic error of the recalibrated score obeys*

$$\big|\mathbb{E}[\hat{f}_t(\hat{r}) \mid \tau] - r(\tau)\big| \;\leq\; c_f \sqrt{\log(2KT/\delta)/t},$$

where K is the number of distinct scores the benchmark produces and the constant c_f is given in Equation (6) *of the proof.*

Proof. We first fix notation for the scores our benchmarks produce. The leaderboard percentile is computed against a finite leaderboard and the VLA reward is binary, so the true scores take finitely many values $\ell_1 < \cdots < \ell_K$, and the recalibration of Equation (4) fits one vertex per level. Write $\Delta_\ell := \max_k(\ell_{k+1} - \ell_k)$ for the largest gap between adjacent levels, $\Delta_\phi := \min_k \big(\phi(\ell_{k+1}) - \phi(\ell_k)\big)$ for the smallest separation the distortion leaves between them, q for the smallest fraction of the anchor pairs that any level receives, and a for the number of new pairs per step. The rate below is informative when $\Delta_\phi > 0$ and $q > 0$, that is, when the distortion collapses no two levels onto one another and no level is starved of anchor pairs, which is the discrete-level analogue of the arm gaps of bandit analysis (Auer et al., 2002) and of the coverage conditions of off-policy evaluation (Chen & Jiang,

2019), and is what turns isotonic regression from a worst-case problem into one with a parametric rate (Zhang, 2002). In terms of these quantities the constant of the statement is

$$c_f \;:=\; \frac{2\Delta_\ell}{\Delta_\phi}\sqrt{\frac{1}{2qa}}. \tag{6}$$

The proof then has three stages that mirror the recalibration pipeline. We first show that the empirical mean of the predictions at each score level concentrates, then that the isotonic projection can only inherit this accuracy, and finally that the interpolated map turns accurate vertices into accurate calibrated scores.

Stage 1, level means concentrate. Coverage is what makes this stage possible, as a level that received no pairs could never be calibrated. At step t each level holds at least qat pairs, each of the form $\hat{r} = \phi(\ell_k) + \xi$ with ξ zero-mean and $|\xi| \le 1$, so Hoeffding's inequality gives $\Pr(|\hat{\mu}_k - \phi(\ell_k)| \ge s) \le 2e^{-2qats^2}$. The guarantee is needed simultaneously for every level and every step, so we take a union bound over the K levels and T steps, and the choice $s = e(t) := \sqrt{\log(2KT/\delta)/(2qat)}$ keeps the total failure probability below δ.

Stage 2, the projection does not hurt. The isotonic fit replaces the raw means $\hat{\mu}_k$ by the closest monotone sequence $\hat{\phi}_k$, and we must check that this projection cannot push an estimate away from the truth. The min–max representation $\hat{\phi}_k = \min_{v \ge k} \max_{u \le k} \operatorname{avg}(\hat{\mu}_u, \dots, \hat{\mu}_v)$ expresses every fitted value as an average of raw means, each within $e(t)$ of its true value. For any $v \ge k$, monotonicity of ϕ gives $\max_{u \le k} \operatorname{avg}[u, v] \ge \operatorname{avg}[k, v] \ge \operatorname{avg}(\phi(\ell_k), \dots, \phi(\ell_v)) - e(t) \ge \phi(\ell_k) - e(t)$, and taking the minimum over v preserves the bound; choosing $v = k$ gives symmetrically $\hat{\phi}_k \le \phi(\ell_k) + e(t)$. So the fitted vertices are as accurate as the raw means.

Stage 3, from vertices to calibrated scores. Separation is what makes this stage possible, as levels the distortion has collapsed onto one another could not be told apart by any monotone fit. When $e(t) \le \Delta_\phi/4$, consecutive fitted vertices stay separated, $\hat{\phi}_{k+1} - \hat{\phi}_k \ge \Delta_\phi - 2e(t) \ge \Delta_\phi/2$, so the piecewise-linear interpolation $\hat{f}_t$ is well defined and its Lipschitz constant is at most $\Delta_\ell/(\Delta_\phi/2) = 2\Delta_\ell/\Delta_\phi$. A trajectory with true score ℓ_k produces predictions with conditional mean $\phi(\ell_k)$, and $\hat{f}_t$ maps the fitted vertex $\hat{\phi}_k$ exactly to ℓ_k, so the systematic error of the calibrated score is at most the Lipschitz constant times the vertex error, $|\hat{f}_t(\phi(\ell_k)) - \ell_k| = |\hat{f}_t(\phi(\ell_k)) - \hat{f}_t(\hat{\phi}_k)| \le \frac{2\Delta_\ell}{\Delta_\phi} e(t)$, which is the claim. □

The last lemma justifies the fusion rule of Equation (5) and supplies the variance V used in the proof of Theorem 4. Its final claim, that the fused variance is flat around the optimal weight, is what Remark 8 invokes to argue robustness to the estimated ratio.

Lemma 12 (Optimal linear fusion)**.** *Let g_1 and g_2 be independent unbiased estimates of the same quantity with positive finite variances V_1 and V_2. Among all their unbiased linear combinations, that is, with weights summing to one, the variance is minimized by weights proportional to V_1^{-1} and V_2^{-1}, and the minimal value is*

$$\Big(\frac{1}{V_1} + \frac{1}{V_2}\Big)^{-1} \;<\; \min(V_1,\, V_2).$$

Moreover, for any weight α on g_1, $\operatorname{Var}(\hat{g}_\alpha) = \operatorname{Var}(\hat{g}_{\alpha^\star}) + (V_1 + V_2)(\alpha - \alpha^\star)^2$, so a misspecified weight costs only a second-order term.

Proof. Write the combination with a single weight, $\hat{g}_\alpha = \alpha g_1 + (1 - \alpha) g_2$, which is unbiased for every α, so the choices differ only in variance. By independence, $\operatorname{Var}(\hat{g}_\alpha) = \alpha^2 V_1 + (1 - \alpha)^2 V_2$, a strictly convex quadratic in α, so the unique minimizer is found by setting the derivative to zero, $\alpha^\star = V_2/(V_1 + V_2)$, that is, weights proportional to the inverse variances. Substituting $\alpha^\star$ back gives $V_1 V_2/(V_1 + V_2)$, which equals the stated harmonic form and is strictly smaller than each of V_1 and V_2 because both precisions are positive. Finally, expanding the quadratic around its minimizer gives $\operatorname{Var}(\hat{g}_\alpha) = \operatorname{Var}(\hat{g}_{\alpha^\star}) + (V_1 + V_2)(\alpha - \alpha^\star)^2$, since its second derivative is $2(V_1 + V_2)$; this is the second-order insensitivity invoked in Remark 8. □

Remark 13 (On restricting to linear combinations)**.** *Lemma 12 optimizes within the linear family, which is the weakest requirement that suits our setting, as it uses only the second moments of the two estimates and preserves unbiasedness whatever their distribution. If the estimates are in addition*

taken to be Gaussian, which group averages approach as the group size grows, the same inverse-variance weights are minimum-variance among all unbiased estimators, so the linear restriction is not what limits the result.

C JUSTIFICATION OF EXPERIMENTAL CHOICES

C.1 KNOWN BENCHMARK LIMITATIONS

Section 5.1 builds the task pool from MLE-Dojo rather than from MLE-Bench directly, for three reasons.

First, part of the original pool is no longer usable. Two of the underlying Kaggle competitions, `detecting-insults-in-social-commentary` and `the-icml-2013-whale-challenge-right-whale-redux`, have been permanently closed, and their official data downloads and submission entries are gone.

Second, the medal-based metric fails to measure model ability. MLE-Bench scores an agent by the number of gold, silver, and bronze medals. At the model scales we study, an agent earns medals on only one or two fixed competitions, so the medal count stays flat no matter how much the agent improves on the rest of the pool. The counts are also skewed, with more golds than silvers or bronzes in published evaluations (Nathani et al., 2025), because a handful of tasks admit near-perfect accuracy and hand out gold medals while the remaining tasks hand out nothing. The ranking against human competitors is what reflects ability, so our protocol reports the leaderboard percentile, which credits improvement on every task.

Third, contamination. The competitions and many of their winning solutions predate the pretraining corpora of current models, a risk acknowledged both in the original release (Chan et al., 2024) and in the accompanying OpenAI blog post (OpenAI, 2024), and the public evaluation has not been actively maintained since.

C.2 DATA SPLITS AND FULL TASK LISTS

MLE-Dojo wraps a superset of the same competitions in an interactive environment with live grading, which avoids the dead entries, exposes leaderboard percentile directly, and allows a free re-partition of the pool. Manually splitting a public pool into train and test portions follows common practice, and our split is fixed once and shared by all configurations.

Why these training and test sets. The partition follows one deterministic rule, fixed before any training, and is designed to keep the categories balanced on both sides. Within each of the three categories (tabular, text, image), the 20 smallest competitions by dataset size are selected so that every task fits the execution sandbox, the 15 smallest of them form the training set, and the remaining 5 form the held-out set. Audio competitions are excluded because they do not fit the sandbox, and one held-out task (`billion-word-imputation`) is dropped because its grader depends on a package unavailable in our environment, leaving 45 training and 14 held-out competitions. The rule has a useful side effect, as every held-out task is larger than every training task of its category, so the held-out evaluation also measures extrapolation from small training tasks to larger unseen ones. We also verified that no task appears on both sides.[3] The data of every competition was cached locally in advance, so training and grading never depend on the Kaggle servers. Table 4 lists the training competitions and Table 5 the held-out ones.

[3] One held-out task shares the Jigsaw competition family with a training task, with different data and a different metric.

Table 4: **The 45 training competitions of MLE-Dojo (train), by category.**

Tabular	Text	Image
mercedes-benz-greener-manufacturing	kaggle-llm-science-exam	aerial-cactus-identification
icr-identify-age-related-conditions	movie-review-sentiment-analysis-kernels-only	leaf-classification
kobe-bryant-shot-selection	llm-detect-ai-generated-text	denoising-dirty-documents
home-data-for-ml-course	random-acts-of-pizza	facial-keypoints-detection
spooky-author-identification	chaii-hindi-and-tamil-question-answering	statoil-iceberg-classifier-challenge
liberty-mutual-group-property-inspection-prediction	nbme-score-clinical-patient-notes	tgs-salt-identification-challenge
detecting-insults-in-social-commentary	20-newsgroups-ciphertext-challenge	global-wheat-detection
walmart-recruiting-store-sales-forecasting	word2vec-nlp-tutorial	whale-categorization-playground
prudential-life-insurance-assessment	jigsaw-toxic-comment-classification-challenge	dog-breed-identification
unimelb	text-normalization-challenge-english-language	plant-pathology-2020-fgvc7
nomad2018-predict-transparent-conductors	wsdm-cup-multilingual-chatbot-arena	dogs-vs-cats-redux-kernels-edition
amazon-employee-access-challenge	text-normalization-challenge-russian-language	petfinder-pawpularity-score
poker-rule-induction	llm-classification-finetuning	plant-seedlings-classification
allstate-purchase-prediction-challenge	stumbleupon	shopee-product-matching
dont-call-me-turkey	linking-writing-processes-to-writing-quality	the-nature-conservancy-fisheries-monitoring

Table 5: **The 14 held-out competitions of MLE-Dojo (test).**

Competition	Category	Task
GiveMeSomeCredit	Tabular	Predict two-year default risk of borrowers
forest-cover-type-kernels-only	Tabular	Predict forest cover type from cartographic features
novozymes-enzyme-stability-prediction	Tabular	Rank thermostability of enzyme variants
integer-sequence-learning	Tabular	Predict the next term of integer sequences
afsis-soil-properties	Tabular	Predict five soil properties from infrared spectra
quora-question-pairs	Text	Decide whether two questions are duplicates
AI4Code	Text	Recover the order of markdown cells in notebooks
jigsaw-unintended-bias-in-toxicity-classification	Text	Detect toxicity while controlling identity bias
quora-insincere-questions-classification	Text	Flag insincere questions
uw-madison-gi-tract-image-segmentation	Image	Segment stomach and bowel in MRI scans
invasive-species-monitoring	Image	Detect an invasive plant species in photos
kuzushiji-recognition	Image	Detect and transcribe cursive Japanese characters
bengaliai-cv19	Image	Classify components of handwritten Bengali graphemes
cassava-leaf-disease-classification	Image	Classify cassava leaf diseases

DSBench. The transfer suite is the data-modeling split of DSBench (Jing et al., 2024), whose tasks are all drawn from Kaggle (74 tasks per the paper, 75 task directories in the released data). We keep the 60 tasks that run end to end in our sandbox. The other 15 are excluded because the training data is too large for the sandbox (seven tasks), the sample submission is missing (eight tasks), or the submission format exceeds the scaffold's column limit (one task). The category columns of Table 1 contain 19 binary classification, 10 multi-class classification, and 23 regression tasks, and the remaining 8 tasks with non-standard metrics form the Other column. We also checked these 60 tasks against the 45 training competitions to ensure no overlap.

C.3 Choice of World Models

A controlled test of the correction mechanisms. Our AutoResearch experiments ask whether Online Debiasing and Inverse-Variance Denoising make an imperfect world model useful for RL. To isolate this effect, the agent and world model start from the same pretrained Qwen3.5 checkpoint at each scale, and the world model remains frozen and is used through prompting alone. No additional simulator training or stronger-teacher supervision is introduced. The ablation in Section 5.5 further holds the two reward streams and their ratio fixed, varying only the active corrections.

Why an external simulator changes the comparison. Dedicated world models can include supervision beyond the agent's initial checkpoint. SWE-World trains Qwen2.5 simulators on execution traces and augments their supervision with reasoning generated by Qwen3-235B-A22B-Thinking (Sun et al., 2026). Qwen-AgentWorld combines continual pretraining on interaction traces and additional domain knowledge with SFT on reasoning-model rollouts and subsequent RL (Zuo et al., 2026). DSWorld trains a Qwen3-8B simulator with SFT and reflective RL, using DeepSeek 3.2 for data synthesis (Yang et al., 2026b). These recipes introduce different sources of additional knowledge or supervision. When their simulated observations enter an agent's trajectory, they can guide later actions; their reward predictions can also guide policy updates. Thus RL and implicit knowledge transfer can occur together, and an improvement alone does not identify how much comes from the correction mechanisms. Our shared-backbone design controls this extra source of supervision. It does not assert that the underlying pretrained checkpoint is free of benchmark exposure, which is a separate issue discussed in Appendix C.1.

Compatibility with other world models. This experimental control is not a requirement of WMRL. The corrections operate on predicted scores and a small stream of paired real-execution scores, independently of how the predictions are produced. With a suitable domain and score interface, simulators of the forms used by SWE-World, Qwen-AgentWorld, and DSWorld can therefore supply the predictions to our framework. Their simulator training and our online reward correction address complementary parts of the problem. This compatibility does not by itself guarantee correction of arbitrary errors in simulated state transitions: our analysis concerns the reward errors under the stated assumptions. The VLA experiment uses the separately pretrained Robometer, illustrating that the framework does not require identical agent and world model architectures. The shared-backbone control above specifically concerns the AutoResearch experiments.

C.4 Choice of Evaluation Tasks

Matching the AutoResearch loop. MLE-Dojo and the data-modeling tasks of DSBench require an agent to propose a solution, implement and execute it on data, inspect the outcome, and revise the solution. The objective is to improve the quality of a learned predictor, measured against other solutions, rather than simply produce executable code. This loop directly instantiates the research process studied in Section 1. SWE-bench emphasizes resolving repository issues, while Terminal-Bench covers a broad range of terminal workflows (Jimenez et al., 2024; Merrill et al., 2026). Both can involve multi-step planning and iterative refinement; our choice is about alignment with data-driven research, not a claim that these benchmarks lack agentic behavior or are easier.

Matching the execution bottleneck. Our target is repeated, data-dependent execution: each candidate research solution can require loading data, training models, and evaluating predictions in its own sandbox. This cost recurs as RL collects more trajectories, which motivates replacing most executions with batched world model predictions. The selected modeling tasks expose this cost together with a measurable quality signal, allowing us to test both speed and final performance. When a patch can be checked by a short CPU test suite, verification need not involve the repeated model training and evaluation that dominate our research workloads. Replacing such inexpensive checks offers limited computational benefit and is not the primary motivation of WMRL. This distinction concerns the cost of obtaining feedback, not the difficulty of producing a correct solution. Execution costs vary across SWE and terminal tasks: complex builds, tests, and environment setup can also be expensive, as SWE-World studies (Sun et al., 2026).

Relation to execution-free software feedback. SWE-RM (Shum et al., 2025) develops learned verifiers to address sparse feedback and limited test coverage, and studies classification accuracy and calibration for software-agent RL. Our motivation is complementary: even when a reliable evaluator is available, obtaining its reward can require running an expensive research experiment. We reduce this execution cost and correct the resulting surrogate errors online. Thus, the relevant distinction is whether environment execution dominates training cost, rather than whether a task has an automatic verifier. We evaluate this regime on MLE-Dojo and DSBench; broader SWE and terminal workflows require their own cost analysis and simulator interface.

D Experimental Details

Training configuration. Both scales train with GRPO, eight groups per step and $n = 8$ trajectories per group, with up to four interaction turns per trajectory, at most 4096 generated tokens per turn, and observations truncated to 1024 tokens. Optimization uses a constant learning rate of 1×10^{-6}, a KL coefficient of 0.04, and an entropy coefficient of 0.002, and rollouts are sampled at temperature 1.0 with top-p 1.0. Real execution runs each solution in an isolated environment with a 1200-second budget. The world model serves the agent backbone with a 12k-token context and returns its prediction within a 1024-token budget, and every step at least one group is graded by real execution to feed the anchor pool.

Sandbox environment. Each solution executes in an isolated sandbox built as a Python 3.11 virtual environment on Linux, with a single ~40 GB NVIDIA GPU and the path contract of the agent prompt, reading inputs from `DATA_DIR` and writing the submission to `SUBMISSION_PATH`. The environment holds 122 packages including transitive dependencies, and installing the directly requested set of Table 6 into a clean virtual environment reproduces it.

Table 6: **Libraries preinstalled in the execution sandbox**, by category, with the exact versions.

Category	Libraries (version)
Scientific computing	numpy 1.26.4, pandas 2.2.3, scipy 1.17.1, statsmodels 0.14.6, sympy 1.14.0, networkx 3.6.1
Classical ML	scikit-learn 1.4.2, xgboost 3.2.0, lightgbm 4.6.0, catboost 1.2.10, joblib 1.5.3
Deep learning (GPU)	torch 2.12.0 (CUDA 13.0), torchvision 0.27.0, tensorflow 2.21.0, keras 3.14.1
NLP	transformers 5.12.1, tokenizers 0.22.2, nltk 3.9.4
Imaging and plotting	Pillow 12.2.0, matplotlib 3.10.9, seaborn 0.13.2, plotly 6.6.0
Utilities	h5py 3.14.0, tqdm 4.67.3, requests 2.34.2, regex

Prompting the world model. The AutoResearch world model is never fine-tuned and works by prompting alone. Its prompt contains the same task description that the agent sees and the agent's current solution, and it is instructed to simulate the execution and report the outcome in the same format as the real environment, from which the score is aggregated. The context is capped at 12k tokens and the prediction at 1024 tokens. Appendix E.1 gives the world model and agent prompts; Appendix C.3 explains the backbone choice.

Recalibration. The monotone map of Equation (4) is the identity until 200 anchor pairs have accumulated, is first fit at that point, and is refit after every 64 new pairs so that the correction follows the drift of the world model.

Calibrating the reference disagreement. We form the disagreement $\hat{\eta}^2$ by normalizing the mean squared residual between calibrated world model scores and true scores on anchor groups by the within-group spread of the true scores, and record its positive warmup value $\hat{\eta}^2_{\mathrm{cal}}$. The implemented rule $\hat{\rho} = 1 + \hat{\eta}^2/\hat{\eta}^2_{\mathrm{cal}}$ compares current disagreement with this reference scale. It is a reward-level proxy for a gradient-level variance ratio, motivated by the second-moment bound in Lemma 10. The warmup calibration sets the proxy scale $c = 1/\hat{\eta}^2_{\mathrm{cal}}$; its calibrated value in one of our actual runs is 0.96, confirming that it is of order one. The normalization gives $\hat{\rho} = 2$ at the reference point by construction, rather than establishing an identity for the true gradient variance ratio. Remark 8 shows that the first-order term of the fusion-variance increase is zero at the optimal weight; the scales measured in our runs put the excess below three percent at the calibration point. These calibration results support the practical approximation at that point, while the sensitivity analysis describes how deviations from the ideal weight affect the fusion objective.

VLA setup. The policy is MiniVLA-1B (Belkhale & Sadigh, 2024), a Prismatic-style model with a Qwen2.5-0.5B language backbone and a vector-quantized action head that encodes an eight-step action chunk into seven discrete tokens, pretrained on LIBERO-90. All models first run supervised fine-tuning on the 50 official demonstrations per task, and then GRPO for 40 steps. Each step samples 64 rollouts per task, and each rollout takes at most 520 environment steps, the benchmark's standard horizon. Training uses the first 16 official initial states of each task while evaluation uses all 50, so most evaluated initial states are never seen in training. Evaluation is greedy, with eight repetitions per initial state, since the simulator's physics is stochastic.

E Prompts

This appendix documents the interfaces of every model we prompt, in both evaluation domains: the AutoResearch agent and its world model in Appendix E.1, and the VLA policy and its world model in Appendix E.2. The AutoResearch prompts are too long to reproduce in full, so we sketch their structure, quote the load-bearing passages verbatim, and mark every omission with [. . .]. The VLA interfaces are short enough to give in full.

E.1 AutoResearch Agent

The agent prompt. Each turn, the AutoResearch agent sees a fixed system prompt, a user prompt holding an automatically generated task overview, and the full history of its previous attempts with their execution feedback. The system prompt consists of the seven segments sketched below. The

user prompt is the first user message of the conversation, as the later turns simply accumulate the interaction history on top of it, and the task overview inside is generated by pure introspection over the task files, with no manual per-task information. Since that overview is what makes the prompt concrete, we instantiate it on one task and carry the same task through the rest of this appendix, namely the Kaggle competition `jigsaw-toxic-comment-classificationchallenge`, a six-label text classification task from our training pool. The sections below are the real ones the builder emits, abridged with [...] where the full text is long.

System prompt of the AutoResearch agent

Role and objective.

```
You are an expert ML engineer competing to MAXIMIZE your leaderboard
Position Score over UP TO K attempts (Position Score = your
percentile rank, higher = better; earned ONLY when your code runs
and writes a VALID submission).  [...]
```

Two-phase strategy.

```
PHASE 1 -- LAND A VALID SUBMISSION FIRST (turn 1):  write a SIMPLE
solution you are confident RUNS end-to-end and writes SUBMISSION_PATH
in the exact required format.  A valid submission is your safety net.
[...]
PHASE 2 -- IMPROVE INCREMENTALLY (later turns):  START FROM YOUR LAST
WORKING CODE, copy it, and change EXACTLY ONE thing per turn.  [...]
If a change ERRORS or scores WORSE, discard it and try a DIFFERENT
single change.  Never rewrite from scratch once something works.
```

Environment contract.

```
ENVIRONMENT CONTRACT (these Python variables are ALREADY defined when
your code runs):  [...]
DATA_DIR: absolute path to the folder with the data files listed
below.  [...]
SUBMISSION_PATH: absolute path to write your submission CSV to.
[...]
A modern NVIDIA GPU (~40 GB) is available.  [...]
Do NOT hardcode '/kaggle/...', '/content/...', or any other path.
```

Sandbox runtime.

```
Runtime:  Python 3.11 on Linux (this is the sandbox where YOUR code
executes).  Installed libraries you may import (name version):  numpy
1.26, pandas 2.2, scikit-learn 1.4, xgboost 3.2, torch 2.12 (CUDA),
transformers 5.12 [...]
Anything NOT in this list is not installed (e.g.  no opencv, no jax).
```

Response format.

```
EACH TURN, respond in TWO parts, in this order:  [...]
1) ANALYSIS (2-4 short sentences of plain text, FIRST): name exactly
what went wrong and the ONE change that fixes it, or the ONE change
you will make to raise the score.  [...]
2) CODE: a SINGLE fenced Python block -- a complete, self-contained
script.
```

Feedback loop.

```
After your code runs, the environment returns your print() output and
-- if a valid submission was written -- your leaderboard Position and
Raw Score; otherwise the full error traceback.  [...]
```

Rules.

```
Keep a valid submission as your fallback; only replace it with code
that RUNS and scores BETTER. [...]
The submission CSV must match the required format EXACTLY. Keep code
efficient so it finishes within the time limit.
```

User prompt of the AutoResearch agent

Competition.

```
jigsaw-toxic-comment-classification-challenge
```

Description (head of the official competition text, capped at 900 tokens; if the cap cuts off the evaluation section, its head is force-appended)**.**

```
Build a multi-headed model that detects different types of toxicity --
toxic, severe_toxic, obscene, threat, insult, and identity_hate -- in
Wikipedia talk-page comments.  [...]
Evaluation:  mean column-wise ROC AUC over the six label columns.
```

Data inventory (every file under `DATA_DIR`, with the full schema of each table)**.**

```
train.csv:  159571 rows × 8 cols:  [id (str), comment_text (str),
toxic (int), severe_toxic (int), obscene (int), threat (int), insult
(int), identity_hate (int)]
    string-column examples -> id:  e.g.  '0000997932d777bf';
comment_text:  e.g.  "Explanation Why the edits made"
test.csv:  153164 rows × 2 cols:  [id (str), comment_text (str)]
sample_submission.csv:  153164 rows × 7 cols:  [id (str), toxic
(float), severe_toxic (float), obscene (float), threat (float),
insult (float), identity_hate (float)]
```

Target columns (present in train but not in test)**.**

```
['toxic', 'severe_toxic', 'obscene', 'threat', 'insult',
'identity_hate']
```

Submission contract (written exactly this way to `SUBMISSION_PATH`)**.**

```
columns = ['id', 'toxic', 'severe_toxic', 'obscene', 'threat',
'insult', 'identity_hate']
FILL the prediction column(s) with your model output:  ['toxic', ...,
'identity_hate'].  COPY the column(s) ['id'] straight from the test
set (id / passthrough).
Example rows (cells truncated for display):
    00001cee341fdb12  0.5  0.5  0.5  0.5  0.5  0.5
    0000247867823ef7  0.5  0.5  0.5  0.5  0.5  0.5
```

Trigger.

```
Begin.
```

The world model prompt. The world model shares one simulator system prompt across eight parallel single-aspect checks, which cover imports and names, API calls against the installed versions, file and column references against the data inventory, shapes and dtypes, the compute budget, remaining runtime errors, submission validity, and solution quality. All eight checks share the same system prompt. The user prompt of each check holds its specific instruction followed by the case block, namely the task overview above and the agent's code with 1-indexed line numbers, and the check returns a verdict in strict JSON. The verdicts are then aggregated into the scalar score $\hat{r}$, where failures cap the score and a valid solution is scored by the quality check. The user prompt below is that of the file and column reference check, instantiated on the same task as above and on a real turn-one attempt at it. That attempt assigns the literal strings `"DATA_DIR"` and `"SUBMISSION_PATH"` to the two variables instead of reading them from the environment, so the first `read_csv` resolves to a path that does not exist. This is exactly the failure the check is meant to catch, and we print the verdict it returns underneath the prompt.

System prompt of the world model

Role.

```
You are a precise execution-and-grading SIMULATOR for Kaggle/MLE
Python solutions.  You do NOT execute code.  You predict -- by
careful static analysis -- exactly what the real sandbox would
report.
```

Environment brief.

```
Sandbox:  Python 3.11 on Linux, ONE ~40GB GPU available.  Installed
(name version):  numpy 1.26, pandas 2.2 [...]
Execution is killed at 1200s.  The code reads data from
os.environ['DATA_DIR'] and MUST write predictions to
os.environ['SUBMISSION_PATH']. [...]
```

Inputs.
```
You are given the TASK OVERVIEW (with a DATA INVENTORY: real files,
columns, row counts, dtypes, and the sample_submission format) and
the agent's CODE (its lines are 1-indexed as shown).  Analyze ONLY
the failure mode this check asks about.
```

Calibration policy.
```
CRITICAL -- DEFAULT TO PASS. Real submissions usually RUN FINE; most
code does NOT trigger this check's failure mode.  [...]
Return verdict='fail' ONLY when you can cite a SPECIFIC line and a
CONCRETE, CERTAIN reason it raises this exact error.  [...]
A false 'fail' on working code is WORSE than a missed error.  When in
doubt, PASS.
```

Error specificity.
```
If you do fail it, be concrete:  identify the EXACT offending source
line, its line number, the precise Python exception class, and the
real error message.  [...]
The agent will READ your env_feedback to fix its code next turn,
so env_feedback MUST look byte-similar to the real sandbox output.
[...]
```

Output format.
```
Output STRICT JSON only, no prose around it:
{"reason":  "<= 2 sentences, the specific cause>", "verdict":
"<pass|fail>", "confidence":  <float 0..1>, "error_type":  [...],
"error_line":  [...], "env_feedback":  [...]}
```

User prompt of the world model

Check instruction.
```
CHECK: Will this code raise FileNotFoundError or KeyError(column) --
i.e.  does it reference a file/path/column that does not match the
DATA INVENTORY? [...]
```

Decision rule.
```
verdict=fail if a FileNotFoundError or KeyError is certain;
env_feedback = the exact message.
```

Task.
```
jigsaw-toxic-comment-classification-challenge
```

Task overview and data inventory (the same overview the agent received, reproduced verbatim)**.**
```
Build a multi-headed model that detects different types of toxicity
[...]
train.csv:  159571 rows × 8 cols:  [id (str), comment_text (str),
toxic (int), ...]  [...]
```

Agent code (the attempt under review, lines 1-indexed)**.**
```
  1  import os
  2  import pandas as pd
  3  import numpy as np
     [...]
  9  # Paths (using the provided environment variables)
 10  DATA_DIR = "DATA_DIR"
 11  SUBMISSION_PATH = "SUBMISSION_PATH"
 12
 13  # Load data
 14  train_df = pd.read_csv(os.path.join(DATA_DIR, "train.csv"))
 15  test_df = pd.read_csv(os.path.join(DATA_DIR, "test.csv"))
     [...]
```

Verdict returned by the world model

```
{
  "reason":  "DATA_DIR is the literal string \"DATA_DIR\" instead of
```

```
os.environ['DATA_DIR'], so read_csv looks for 'DATA_DIR/train.csv'
which doesn't exist.",
  "verdict":  "fail",
  "confidence":  0.95,
  "error_type":  "FileNotFoundError",
  "error_line":  14,
  "env_feedback":  "=== Code Execution Results ===\nExecution
failed:  Traceback (most recent call last):\n  File
\"/tmp/tmph32t8gzr.py\", line 28, in <module>\n    train_df =
pd.read_csv(os.path.join(DATA_DIR, \"train.csv\")) [...]"
}
```


E.2 VLA AGENT

The agent prompt. At every control step the VLA agent is prompted from scratch, as the policy is memoryless and no interaction history accumulates on top of the first message. MiniVLA-1B is built on the `openvla-mini` (Prismatic) codebase and we reuse its prompt builder unchanged, so the prompt is the stock Qwen-2 chat template of the backbone, with one system turn and one user turn. We add no prompt engineering of our own: the system turn is the stock backbone line, and the user turn carries only the current camera image and the task instruction, lower-cased. Keeping the template byte-identical to the upstream one matters because the checkpoint we start from is pretrained behind exactly this template, so any deviation in wording, casing, or the chat scaffold would move the policy off its pretraining distribution before RL even starts. We instantiate it below on task 8 of LIBERO-Long, *put both moka pots on the stove*, and carry that task through the rest of this subsection.

System prompt of the VLA agent

Role (the stock backbone line, unchanged)**.**

```
<|im_start|>system
You are Qwen, created by Alibaba Cloud.  You are a helpful
assistant.<|im_end|>
```

User prompt of the VLA agent

Visual observation (the third-person agentview frame the simulator renders at the current step, resized to 224×224; it enters the sequence as projected patch embeddings, not as text)**.**



Instruction.

```
<|im_start|>user
What action should the robot take to put both moka pots on the
stove?<|im_end|>
<|im_start|>assistant
```

Response format.

```
<exactly 7 discrete action tokens, one code from each of the
7 residual codebooks of the action head; no free-form text is
generated>
```

The world model prompt. The world model of this domain is Robometer-4B, a full fine-tune of Qwen3-VL-4B-Instruct on robot-progress data. Unlike the AutoResearch world model it is never asked to simulate an execution it has not run: it reads the frames the policy actually produced and estimates how far the task has progressed at each of them. It also carries no system prompt at all. One rollout becomes one single-turn conversation, whose user message holds the instruction text below followed by the frames the policy observed, uniformly subsampled to eight, the window the model was trained on. The reply is a per-frame progress value together with a per-frame success probability.

User prompt of the world model

Instruction (verbatim; the task string is inserted)**.**

```
The task for the robot is 'put both moka pots on the stove'.  Given
the trajectory video, predict the task progress at each frame, how
far along the robot is towards completing the task, a float between
0 and 1, where 0 is the starting state and 1 is when the task is
completed.  If the robot is not performing the same task, predict
0 progress.
```

Rollout frames (8 frames uniformly subsampled from the rollout; 4 of them shown, spanning start to end of the episode)**.**



Response format.

```
<a progress value in [0,1] and a success probability in [0,1] for
each of the 8 frames, read out by the discrete progress head and the
success head>
```