Feedback-Driven Scaling for Reasoning and Generation

Nikash Bhardwaj

May 2026

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:

Deepak Pathak

Thesis submitted in partial fulfillment of the requirements for the SCS Undergraduate Research Thesis.

Keywords: LLM, Reasoning, Reinforcement Learning, Sim-to-real, Simulator, Text-to-image, Iterative Refinement, Inference, Generation

Abstract

This work studies two complementary ways to improve reasoning and generation in foundation models through feedback-driven scaling. The first focuses on large language models in scientific domains, where internet-scale question-answer data is limited, especially for physics. To address this, a physics simulator is used to generate random physical scenes, derive synthetic question-answer pairs from simulated interactions, and provide scalable training data for reinforcement learning. Models trained only on this synthetic simulator data show strong sim-to-real transfer on evaluation settings, improving performance on International Physics Olympiad problems by roughly 5 to 10 percentage points across model sizes, which suggests that simulators can serve as a practical source of supervision for domains with scarce natural training data. The second focuses on text-to-image generation, where current models often struggle with compositional prompts containing multiple objects, attributes, and relationships. Rather than relying only on more denoising steps or parallel sampling, this work treats generation as an iterative refinement process in which an initial image is repeatedly revised using feedback from a vision-language model critic. This lightweight test-time strategy improves prompt fidelity across several compositional benchmarks, including gains of 16.9% on ConceptMix, 13.8% on the 3D-Spatial portion of T2I-CompBench, and 12.5% on Visual Jenga relative to a compute-matched baseline, with human evaluations also preferring the iterative approach. Taken together, these results show that explicit feedback, whether through simulator-based reinforcement learning or inference-time iterative refinement, provides an effective path toward stronger reasoning and more reliable generation.

Acknowledgments

To begin, I would like to thank my advisor, Deepak Pathak, for his support in allowing me to work in his lab. In the lab, I would like to acknowledge the time that Mihir Prabhudesai and Shantanu Jaiswal have spent with me in sharing their research expertise. To all the other members of the lab, I am grateful for the time I spent in the group and your advice.

At Carnegie Mellon, I would also like to acknowledge William Hrusa for guiding me through my first research experience and David Touretzky for introducing me to AI research through a SURF project. Your insights and advice helped me to become a better researcher. I would also like to thank my advisor, Thomas Cortina, for helping me throughout my time in SCS. To all my other professors and friends, you each helped to shape my growth to do this work.

Finally, I am grateful to my entire family for the support you have provided over the years and for the sacrifices you have made to help me throughout my journey. This work is as much a culmination of your effort and dedication as it is of mine.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most efforts to improve large language models focus on scaling along three familiar axes: increasing parameter count, expanding pretraining corpora, and allocating more compute to support larger models and datasets. This work instead emphasizes two complementary routes to stronger reasoning: richer supervision during post-training and more structured computation at inference time. In large language models, reinforcement learning with verifiable rewards (RLVR) shows that objective feedback can push models beyond shallow pattern matching toward more reliable multi-step reasoning. At the same time, work on chain-of-thought prompting and related test-time strategies shows that performance can also improve when models are allowed to generate intermediate steps, inspect partial solutions, and refine their outputs before producing a final answer. These developments suggest a broader view of progress in reasoning for LLM systems. Stronger reasoning does not arise only from a larger-is-better approach to parameters, data, and compute; it also depends on whether the model can receive informative feedback, either across post-training episodes or within the generation process itself.

In Chapter 2, we see this perspective is especially important in physics and other scientific disciplines, where the primary bottleneck is often the lack of scalable supervision data. Unlike many language tasks that can draw on massive web-scale corpora, scientific reasoning problems are limited in quantity, narrow in topical coverage, and uneven in difficulty. Physics illustrates this challenge particularly well. Solving physics problems often requires numerical reasoning, hidden-variable inference, symbolic manipulation, and counterfactual analysis, yet publicly available datasets capture only a small and incomplete subset of these capabilities. If reinforcement learning is to scale effectively in such settings, it cannot rely solely on fixed collections of textbook or internet problems. A more promising alternative is to use simulators as generators of training data. Because simulators encode domain laws in executable form, they can produce large numbers of grounded trajectories together with

precise state information and other supervisory signals that are difficult to obtain from text alone. This makes it possible to construct verified reasoning examples at scale rather than treating supervision data as an inherently scarce resource. In turn, reinforcement learning can shift from memorizing a limited corpus to learning from a renewable stream of grounded examples. More broadly, this provides a pathway for extending reasoning-oriented training to domains where naturally occurring data are sparse but the underlying structure is sufficiently well understood to support simulation.

In Chapter 3, a complementary line of work asks not how to produce better training signals, but how to use computation more effectively at inference time. In language models, chain-of-thought prompting shows that performance often improves when models reason through intermediate steps rather than attempting a one-shot answer. In image generation, a similar challenge appears in compositional tasks, where a model must satisfy many interacting constraints simultaneously. Standard one-pass generation or parallel sampling may increase diversity, but these approaches do not allow the system to preserve partial successes and correct specific errors in the generation process over time. Iterative refinement offers an alternative. An image generator first produces an initial candidate for a prompt, a vision-language critic identifies errors relative to that prompt, an editor proposes targeted revisions for the image generator, and a verifier evaluates the updated result. This refinement process introduces a form of self-correction that is analogous in spirit to intermediate reasoning in language models: instead of requiring the full solution to emerge in a single pass, the system can improve it progressively, step by step.

The connection between these two directions is that simulator-based RLVR and iterative refinement both replace weak one-shot supervision with explicit feedback loops. In one case, the loop operates during post-training, where simulator-derived data and verifiable rewards shape the model's parameters. In the other, the loop operates during inference, where critics, editors, and intermediate reasoning improve a candidate output before it is finalized. Both approaches therefore address the same underlying problem: how to build large language models that can continue to improve reasoning in domains where traditional scaling methods are insufficient. For scientific reasoning, simulators provide a practical way to generate trustworthy supervision at scale, bridging the sim-to-real gap. For complex reasoning tasks and generation more broadly, iterative refinement provides a mechanism for turning feedback into better solutions. Taken together, these ideas suggest that the next stage of progress in reasoning systems may come not only from larger pretraining pipelines, but from post-training and inference methods explicitly designed to leverage simulated data and support critique, correction, and structured improvement.

Chapter 2

Improving Physical Reasoning in LLMs using Reinforcement Learning with Physics Simulator Data

2.1 Abstract

Recent advances in large language model reasoning in models like DeepSeek-R1 have been driven largely by the widespread availability of question-answer data from the internet, but this resource is limited in both scale and domain coverage. This limitation is especially apparent in physics, where large datasets for training reasoning-capable models are far less available than in fields such as mathematics. In this work [27], we investigate physics simulators as an alternative and scalable source of data by generating random physical scenes, constructing synthetic question-answer pairs from simulated interactions, and training language models on this data with reinforcement learning. Using this approach, we show that models trained only on synthetic simulator data can transfer effectively to real-world evaluation settings, improving performance on International Physics Olympiad problems by roughly 5 to 10 percentage points across different model sizes. These findings suggest that physics simulators can provide a practical path for developing stronger physical reasoning in language models beyond the limits of internet-derived datasets.

2.2 Introduction and Related Work

Recent advances in large language models have shown that reinforcement learning with verifiable rewards (RLVR) can push these systems beyond simple pattern recognition and

toward genuine multi-step reasoning. However, this progress depends on the availability of high-quality question-answer supervision, and that dependency creates an important limitation. Existing training corpora drawn from textbooks and internet sources are finite, unevenly distributed across topics, and difficult to scale beyond a few million examples. For this reason, recent reasoning-oriented systems such as RLVR-based DeepSeek-R1 are constrained less by model size or optimization capacity than by the shortage of reliable supervision data itself [8, 40]. In other words, as reasoning models improve, the central challenge increasingly becomes how to generate enough high-quality training signals to continue that improvement.

This limitation is particularly severe in the physical sciences. In mathematics, large collections of question-answer pairs are widely available, making it easier to train and evaluate reasoning models. Physics, chemistry, and related scientific fields do not have the same level of dataset coverage, especially for problems that require structured, multi-step reasoning. Fewer than 1 percent of the roughly 800,000 question-answer pairs used in DeepSeek-R1 involve STEM content, which helps explain why performance on standard physics benchmarks remains relatively weak. More broadly, internet-based physics questions are not only limited in quantity, but also sparse in their coverage of concepts, inconsistent in form, and insufficiently varied in difficulty. As a result, they fail to provide the systematic supervision needed for models to learn broad and transferable scientific reasoning.

Physics engines present a compelling alternative because they encode physical laws in executable form rather than in written descriptions. Instead of summarizing phenomena in text, these simulators calculate how a system evolves over time by numerically integrating ordinary differential equations under physical constraints. This allows them to produce a potentially unlimited number of trajectories together with detailed supervision signals such as forces, momentum changes, and energy transfer. At the same time, there is a major gap between raw simulator output and the kind of reasoning required to solve physics problems. Simulators typically produce approximate, continuous, forward-time numerical traces, whereas many physics tasks demand inverse reasoning, symbolic derivation, and counterfactual analysis. One possible response is to use simulators as external tools that language models can call directly, but this introduces a new problem: the model must now generate correct code and use simulator-specific APIs effectively, which is itself a difficult task [28, 29]. These papers argue that this tool-use approach is hard to scale, partly because models often fail to produce executable and physically accurate code, and partly because many physical effects are not natively implemented in simulators and would require additional manual engineering.

To address these issues, we propose Sim2Reason, a framework that turns a physics simulator into a scalable generator of verified question-answer data as shown in Figure 2.1. Rather than relying on the model to write simulation code, the method procedurally constructs diverse physical systems inside the simulator and then automatically converts simulated behavior into training examples. These examples are designed to cover three forms of reasoning: numeric questions that ask about system states, reverse questions that require inferring hidden parameters, and symbolic questions that involve closed-form expressions. Using a domain-specific language (DSL), the framework can combine different mechanics concepts into more complex scenes, such as systems involving pulleys together with rotational motion, and in this way generate millions of distinct training samples grounded in classical mechanics. Filtering is also done to improve question quality, since simulator-generated problems can otherwise become either trivial or computationally impractical; pruning these low-value cases helps focus training on examples that fall within a productive difficulty range. The models are then trained with reinforcement learning using only this synthetic data during post-training, with no real-world physics question-answer pairs added at this stage. Evaluations on IPhO, JEEBench, PHYSICS, and OlympiadBench show consistent improvements, including gains of about 5 to 10 percentage points on zero-shot IPhO mechanics problems across models ranging from 3B to 32B parameters, as well as a reported 17.9 percent gain on JEEBench for 32B models. Taken together, these results suggest that Sim2Reason is not simply teaching models to imitate simulator behavior, but is helping them develop a more general capacity for physical reasoning while also providing a scalable basis for benchmarking foundation models.

2.3 Method

Our method trains large language models for physical reasoning by first creating synthetic supervision from a physics simulator and then using that data for post-training. The simulation platform is MuJoCo [32], and the design generates data to cover a broad range of physics topics, including kinematics, rotational motion, orbital dynamics, variable-mass systems, and basic electromagnetism. The overall pipeline has four main stages: scene generation, physics simulation, question-answer construction, and data filtration. The central idea is to transform simulator outputs into verified reasoning examples that can support reinforcement learning with verifiable rewards in four steps as shown in Figure 2.2: Scene Generation, Physics Simulation, Question-Answer Pair Construction, and Data Filtration.

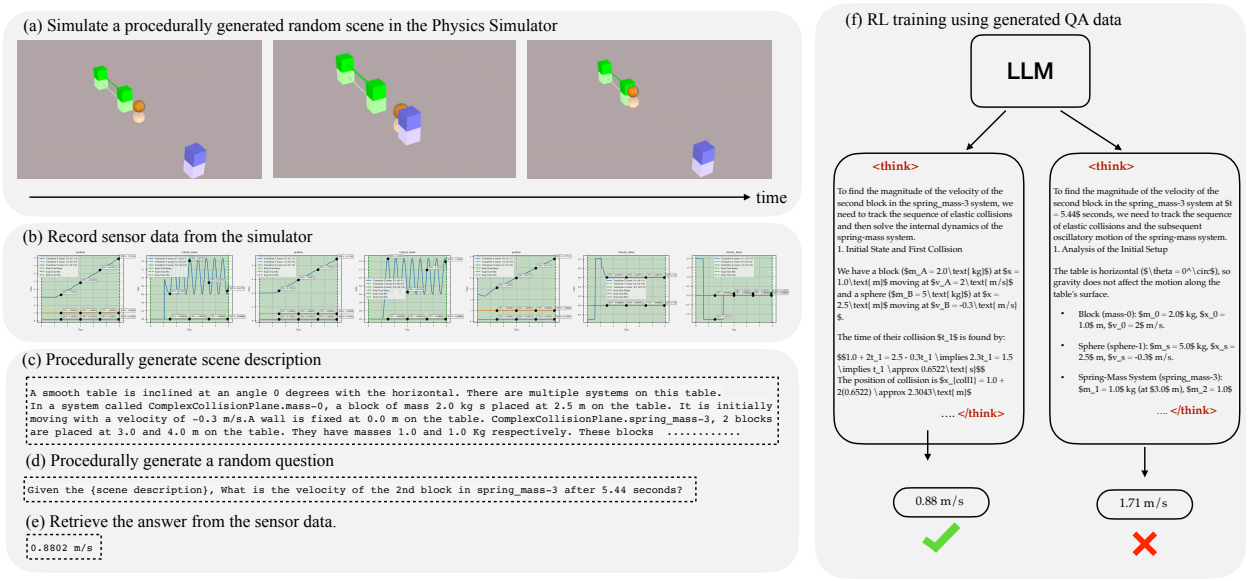

Figure 2.1: Overview of the SIM2REASON pipeline.

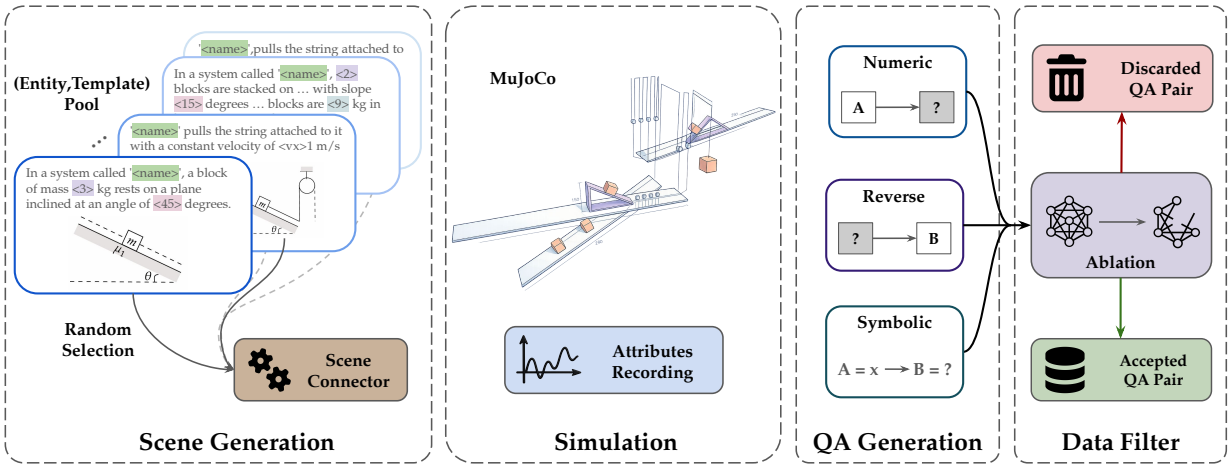

Figure 2.2: Overview of the synthetic data-generation pipeline.

To generate scenes at scale, the domain-specific language separates meaningful physical variations from superficial ones. For example, changing a block's mass may alter the reasoning required to solve a problem, whereas changing a pulley string's length may not significantly affect the relevant dynamics. The DSL has three abstraction levels: the **body**, the **entity**, and the **scene**. A body is the basic physical object, such as a block or sphere, and each body has a name, type-specific parameters, a MuJoCo XML template, and a cor-

responding natural-language description. Bodies are then assembled into entities, which are predefined physically valid structures with specific connection rules; this prevents invalid constructions such as attaching objects in ways that do not make physical sense. Finally, full scenes are created by randomly selecting entities and connecting them, after which the full MuJoCo XML is built by composing the XML templates of the chosen entities and bodies. This structured design allows the system to produce large numbers of simulatable scenes without human intervention.

Once a scene is built, it is simulated in MuJoCo and the system records important physical quantities over time for each body. The recorded quantities depend on the object type: for masses, the system tracks dynamics variables, while for strings it records properties such as tension and length. Because simulator traces can contain abrupt or poorly modeled transitions, such as collisions or bodies falling off surfaces, a pruning rule is applied to remove unstable time segments. A sliding-window mean and standard deviation of acceleration, written as

$$\mu_t = \text{mean}\{a_j\}_{j=t}^{t+w}, \qquad \sigma_t = \text{std}\{a_j\}_{j=t}^{t+w}, \tag{2.1}$$

are computed, and we then truncate the trace at time t if

$$\max_{i \in \{t,\ldots,t+w\}} |a_i - \mu_t| \geq k\sigma_t. \tag{2.2}$$

Here, a denotes the recorded acceleration and k controls the sensitivity of spike detection; a value of $k = 5$ controls the sensitivity during actual data generation. In addition, the simulator is extended to handle variable-mass systems, Newtonian gravity, and collisions with a chosen coefficient of restitution.

After simulation, the recorded trajectories are converted into natural-language question-answer pairs. First, the descriptions of the entities and their interconnections are combined to generate a textual scene description. Next, a body, a physical quantity, and a time step are sampled in order to build one of three question types. **Numeric questions** test forward reasoning by asking for a physical quantity at a particular time, such as a velocity after several seconds. **Reverse questions** test inverse reasoning by hiding one scene parameter and asking the model to infer it from observed behavior. **Symbolic questions** replace numerical parameters with symbols and require the model to reason in a more algebraic form, for example by expressing velocity as a function of time. This three-part design moves beyond simple state lookup and encourages multiple styles of physical reasoning.

The filtration stage is designed to remove degenerate questions, especially those that admit shortcut solutions. A shortcut arises when the correct numerical answer can still be obtained even if the model ignores part of the scene or replaces a multi-body interaction with an oversimplified approximation as shown in Figure 2.3. The correct answer depends on the coupled motion of the block m and wedge M, but weaker models may collapse the dotted region into a single body of mass $M + m$ and still match the numeric answer. We filter QA pairs whose answers are invariant to such approximations. To detect these shortcut solutions, controlled ablations of each original scene are created. In **entity-removal ablations**, the scene is treated as a graph, one entity is removed at a time while preserving connectivity, and then the resulting sub-scene is re-simulated. In **joint-removal ablations**, selected joints or constraints are replaced with rigidly glued components. If a question's ground-truth answer remains unchanged in any ablated version, the question is discarded. This ensures that accepted examples genuinely depend on the full physical structure of the scene.

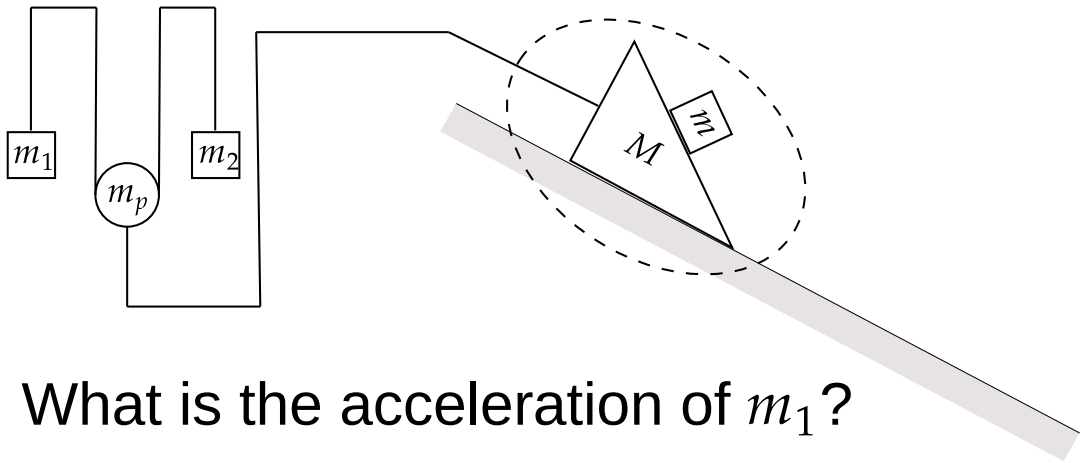

Figure 2.3: Illustration of shortcut detection through scene ablation.

The final stage uses reinforcement learning with verifiable rewards to post-train the language model on the filtered synthetic dataset. For each prompt x, the policy $\pi_\theta(\cdot \mid x)$ samples a group of G responses $\{y_i\}_{i=1}^{G}$, and each response receives a scalar reward $R(x, y_i)$ based on whether the final answer is exactly correct. Training is performed with Group Sequence Policy Optimization (GSPO) [49], using the base instruction-tuned model as a reference policy π_{ref}. As in other group-based RL methods, the rewards are normalized within each group to produce group-relative advantages. The GSPO objective is

$$L_{\mathrm{GSPO}}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\min\Big(\rho_i\hat{A}_i,\ \mathrm{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)\hat{A}_i\Big)\right], \tag{2.3}$$

where

$$\rho_i = \frac{\pi_\theta(y_i \mid x)}{\pi_{\mathrm{ref}}(y_i \mid x)}. \tag{2.4}$$

To improve efficiency under sparse rewards, DAPO-style Dynamic sAmpling Policy Optimization [45] is adopted: if the reward standard deviation within a group is near zero, meaning the group provides little learning signal, the system resamples additional prompts so that training batches contain more informative examples.

2.4 Experiments

The experimental evaluation studies whether simulator-generated supervision can improve real-world physical reasoning after post-training. The setup applies reinforcement learning on synthetic data produced by the SIM2REASON pipeline and then measures transfer to held-out synthetic questions and external benchmarks. Unless otherwise noted, post-training uses the **numeric** QA format described in 2.3, with questions generated on the fly from the simulator. Each run lasts for 200 RL steps with batch size 32, so the model sees approximately 6,400 distinct question-answer pairs during post-training. The evaluation suite spans both in-domain and out-of-domain reasoning tasks:

- **International Physics Olympiad (IPhO):** a curated zero-shot mechanics set built from problems spanning 1967–2025 [15], with 77 filtered questions; for problems containing diagrams, figure captions are generated from the original problem context using GPT-4o.
- **HCV (Concepts of Physics):** a 512-question mechanics evaluation set curated from H. C. Verma's *Concepts of Physics*, again supplemented with GPT-4o-generated figure captions when diagrams appear [33].
- **JEEBench:** a 515-problem benchmark drawn from JEE-Advanced and covering physics, chemistry, and mathematics; evaluation is restricted to text-only mechanics questions and follows the official pipeline [2].

- **OlympiadBench:** a difficult STEM benchmark composed of international and national olympiad problems; only text-only mechanics questions are used when applicable, and exact-match accuracy is reported following [11].
- **PHYSICS:** a textbook-derived benchmark with publicly released test data; evaluation is limited to mechanics-related, text-only questions using the official setup from [50].
- **AIME 2025:** an out-of-domain mathematics benchmark used to test whether gains transfer beyond physics; evaluation uses the LightEval pipeline and reports mean@8 (mean accuracy over 8 sampled responses) [1, 10].
- **MATH 500:** a 500-problem subset of the Hendrycks MATH benchmark, consisting of competition-style problems with numeric or symbolic final answers, evaluated by exact match [12].

The model suite includes Qwen2.5 Instruct checkpoints at 3B, 7B, 14B, and 32B parameters, together with Qwen3-30B-Instruct as a stronger baseline. A notable experimental detail is the large difference in response length across families: Qwen3-30B typically produces responses of about $\sim$ 8k tokens, whereas similarly sized Qwen2.5 models average only $\sim$ 1.5k tokens. Because this substantially increases reinforcement-learning cost, Qwen3-30B is trained for only 100 RL steps, while the Qwen2.5 models are trained for 200 steps. Figure 2.4 tracks the Qwen3-30B-Instruct run and shows that longer responses are strongly associated with higher synthetic validation accuracy: validation accuracy rises during post-training while average response length grows into the 6.5k–8.5k token range. This trend suggests that the post-training process encourages more extensive intermediate reasoning rather than simply changing final-answer behavior.

The main zero-shot transfer results are reported in Table 2.1 and indicate that RL on synthetic mechanics questions improves IPhO performance across all tested model sizes, even though no real-world physics QA pairs are used during post-training. Qwen3-30B improves from 35.6% to 40.0% on IPhO Mechanics, a gain of +4.4 points, while HCV rises from 53.9% to 59.0%. The Qwen2.5 family shows similarly consistent gains: Qwen2.5-32B improves from 19.8% to 25.2% on IPhO, Qwen2.5-14B from 16.07% to 20.45%, Qwen2.5-7B from 10.7% to 15.1%, and Qwen2.5-3B reaches 13.15% after RL, corresponding to a reported gain of +7.5 points. On the held-out synthetic evaluation splits, improvements are especially large for numeric questions; for example, Qwen2.5-32B increases from 8.9% to 21.9% on synthetic numeric evaluation, and Qwen2.5-7B rises from 7.7% to 16.3%. Table 2.1 also shows that post-training on numeric questions transfers to other reasoning modes, since

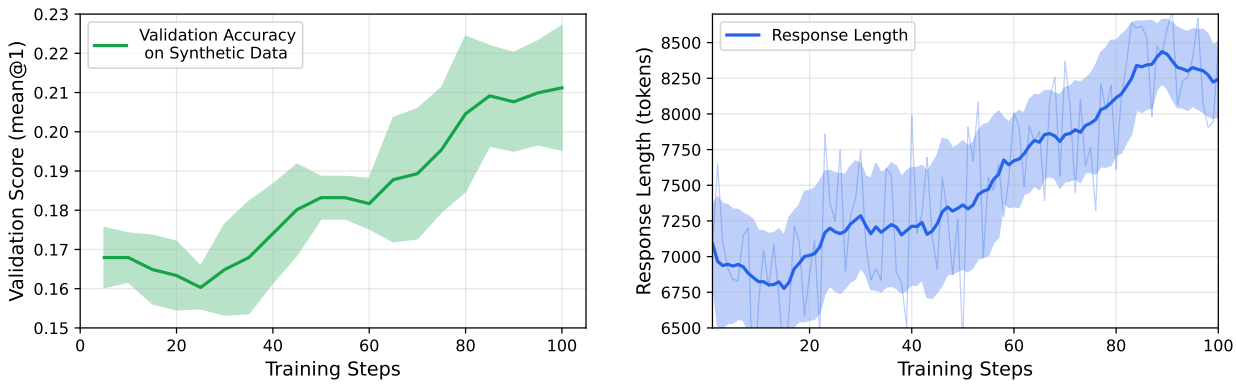

Figure 2.4: Average response length and synthetic validation accuracy for Qwen3-30B-Instruct during RL post-training.

synthetic symbolic accuracy improves for every Qwen2.5 model, indicating that the learned signal is not tied only to one prompt template.

Model	Synthetic Numeric	Synthetic Symbolic	HCV	IPhO Mechanics
Qwen3-30B	14.8%	8.8%	53.9%	35.6%
+ RL (synthetic)	17.4% (+2.6%)	8.0% (-0.8%)	59.0% (+5.1%)	40.0% (+4.4%)
Qwen2.5-32B	8.9%	5.6%	50.6%	19.8%
+ RL (synthetic)	21.9% (+13.0%)	10.4% (+4.8%)	53.9% (+3.3%)	25.2% (+5.4%)
Qwen2.5-14B	7.0%	5.6%	49.3%	16.07%
+ RL (synthetic)	17.0% (+10.0%)	10.4% (+4.8%)	51.7% (+2.4%)	20.45% (+4.4%)
Qwen2.5-7B	7.7%	5.6%	44.5%	10.7%
+ RL (synthetic)	16.3% (+8.6%)	9.6% (+4.0%)	46.3% (+1.8%)	15.1% (+4.4%)
Qwen2.5-3B	4.8%	3.2%	31.9%	5.68%
+ RL (synthetic)	12.5% (+7.7%)	9.4% (+6.2%)	39.5% (+7.6%)	13.15% (+7.5%)

Table 2.1: Performance of Qwen2.5 family Instruct models and Qwen3-30B before and after RL on synthetic datasets, expressed in percentage. Improvements are shown in parentheses.

The broader transfer pattern is shown in Table 2.2 using Qwen2.5-32B on additional real-world benchmarks. The largest improvement appears on JEEBench, where performance increases from 34.38% to 52.28%, a gain of +17.90 points. PHYSICS improves from 39.42% to 43.09%, OlympiadBench from 41.41% to 44.53%, AIME 2025 from 10.83% to 12.5%, and MATH 500 from 78.4% to 82.8%. These gains show that the effect is not limited to the simulator distribution itself: the strongest increases occur on mechanics-heavy benchmarks that align closely with the simulated phenomena, but improvements also extend to out-of-domain mathematics tasks, suggesting stronger algebraic manipulation and multi-step quantitative reasoning. In addition to serving as a training source, the framework is also used as a scalable benchmarking tool. Because simulator-generated questions come with automatically verifiable answers and can be produced in large quantities, the same pipeline supports rapid evaluation and fine-grained diagnosis across specific physical phenomena in a way that is difficult to achieve with small, manually curated real-world benchmarks.

The comparison between post-training strategies in Table 2.3 shows that reinforcement learning with verifiable rewards is substantially more effective than supervised fine-tuning for this setting. For the 32B model, the baseline achieves 14.0% on the synthetic evaluation and 19.8% on IPhO. Supervised fine-tuning on 200,000 rejection-sampled teacher trajectories from GPT-4, o3, and o4-mini improves synthetic performance only modestly, from 14.0% to 16.0%, and reduces IPhO accuracy from 19.8% to 15.9%. By contrast, RL raises synthetic performance to 32.0% and IPhO performance to 25.2%. This difference suggests that sparse outcome-based optimization is better suited to distilling simulator-derived supervision while preserving general reasoning ability, whereas aggressive imitation-style updates on a narrow

Benchmark	Model	Score
JEEBench	Qwen2.5 32B	34.38%
	+ RL (synthetic)	52.28%(+17.90%)
PHYSICS	Qwen2.5 32B	39.42%
	+ RL (synthetic)	43.09%(+3.67%)
OlympiadBench	Qwen2.5 32B	41.41%
	+ RL (synthetic)	44.53%(+3.12%)
AIME 25	Qwen2.5 32B	10.83%
	+ RL (synthetic)	12.5%(+1.67%)
MATH 500	Qwen2.5 32B	78.4%
	+ RL (synthetic)	82.8%(+4.4%)

Table 2.2: Mean accuracy of Qwen 2.5 32B Instruct on other real world benchmarks.

distribution can introduce a harmful shift away from the base instruction-following model, consistent with observations about catastrophic forgetting in post-training [30].

Model (Qwen 2.5 32B)	Synthetic	IPhO Mechanics
Instruct Base	14.0%	19.8%
+ SFT (Rejection Sampling)	16.0% (+2.0%)	15.9% (-3.9%)
+ RL (Ours)	**32.0%** (+18.0%)	**25.2%** (+5.4%)

Table 2.3: Comparison of RL vs. SFT on 32B model performance.

The ablation studies in Table 2.4 clarify which design choices matter most. In the QA-format ablation, reported on Qwen2.5-3B Instruct using IPhO Mechanics, the baseline starts at 5.68%. RL on **reverse** questions produces only a small increase to 5.84%, RL on **symbolic** questions reaches 7.46%, and RL on **numeric** questions reaches 13.15%, making numeric supervision the strongest training signal for transfer to real-world mechanics. The data-filtration ablation in Table 2.4(b) shows that removing shortcut-solvable questions is also crucial. Without filtering, IPhO rises only from 5.68% to 7.14%, whereas filtered training reaches 13.15%. This result supports the claim that degenerate questions can reward physically incomplete reasoning and that scene-ablation filtering produces a much cleaner supervision distribution.

A further comparison is made against DAPO-17K, a public math RL dataset released with the DAPO system [45], because no comparable large-scale public physics post-training

(a) Improvements by each QA format during RL post-training.

Model (Qwen 3B)	IPhO
Baseline	5.68%
+ RL (reverse)	5.84%
+ RL (symbolic)	7.46%
+ RL (numeric)	**13.15%**

(b) Effect of shortcut-solution filtering.

Model (Qwen 3B)	IPhO
Baseline	5.68%
+ RL (no filter)	7.14%
+ RL (filtered)	**13.15%**

Table 2.4: Ablations on (a) QA format and (b) Data filtration.

dataset is currently available. Table 2.5 shows that for Qwen 3B, the baseline IPhO score is 5.68. Training on DAPO-17K alone raises this to 9.98, and mixing DAPO-17K with synthetic simulator data increases it slightly further to 10.35. However, training on the SIM2REASON synthetic data alone yields 13.15, despite this ablation using only a 1K-sample synthetic subset, which is much smaller than the 17K-example real dataset. This provides evidence that domain-aligned simulator supervision carries a higher-quality learning signal for physics reasoning than a substantially larger but less targeted mathematical corpus. The simulator is therefore useful not only as a generator of post-training data, but also as a controlled environment for comparing alternative data sources and training strategies under matched evaluation conditions.

Model (Qwen 3B)	IPhO
Baseline	5.68
+ RL DAPO-17K (Real)	9.98
+ RL Mixed: DAPO-17K (Real) + Synthetic	10.35
+ RL Synthetic (Ours)	**13.15**

Table 2.5: Comparison of data sources for RL post-training on Qwen 3B. Synthetic simulator data alone outperforms the larger DAPO-17K real dataset.

The final experimental analyses examine difficulty scaling, generalization beyond directly simulated scenarios, and qualitative changes in reasoning behavior. Table 2.6 evaluates Qwen 32B on PHYSICS across difficulty tiers and shows gains at every level after RL on synthetic data: *High School and Below* rises from 65.5% to 68.3% (+2.8), *High School Olympiad* from 52.9% to 54.0% (+1.1), *Undergraduate* from 47.9% to 48.4% (+0.5), and *Postgraduate* from 32.2% to 37.8% (+5.6). The largest gain at the postgraduate level indicates that simulator-based RL is especially helpful for harder multi-step problems. Generalization also

extends beyond situations explicitly modeled in MuJoCo. Figure 2.5 presents a rocket-escape problem involving both Earth and Sun, which would require additional bespoke entity design to simulate directly. In eight trials, the base Qwen2.5-32B-Instruct model fails every time, whereas the RL-finetuned model solves the problem correctly in 4/8 trials, using energy-based reasoning with $v_{\text{total}} = \sqrt{v_{\text{sun}}^2 + v_{\text{earth}}^2}$ rather than incorrect linear addition. Qualitative examples are included in 2.6 (Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, and Figure 2.10) show improvements along three recurring axes: reduced arithmetic mistakes, better mapping from verbal problem statements to the correct equations and boundary conditions, and stronger strategic planning through intermediate checks and unit conversions. Together, these experiments support the conclusion that simulator-derived RL improves not only in-distribution performance but also transferable scientific reasoning behavior on real-world problems.

Category	**Qwen2.5 32B Instruct**	**+ RL (synthetic data)**
High School and Below (HSB)	65.5%	68.3% (+2.8%)
High School Olympiad (HSO)	52.9%	54.0% (+1.1%)
Undergraduate (UG)	47.9%	48.4% (+0.5%)
Postgraduate (PG)	32.2%	**37.8%** (+5.6%)

Table 2.6: Detailed performance across difficulty levels on the PHYSICS benchmark.

2.5 Conclusion

SIM2REASON is a simulator-driven framework for physical reasoning that procedurally generates diverse physics scenes, converts simulation traces into verifiable question–answer pairs, and post-trains large language models using reinforcement learning with verifiable rewards (RLVR). Synthetic supervision alone produces consistent zero-shot sim-to-real gains on real-world benchmarks such as IPhO mechanics, indicating that physics simulators can serve as a scalable source of reasoning supervision when naturally occurring QA datasets are limited. A natural extension is to broaden the framework beyond classical mechanics to areas such as electromagnetism, thermodynamics, and other physical sciences. A major potential benefit is improved access to high-quality scientific tutoring and problem-solving tools, together with reduced reliance on internet-scraped QA corpora. In summary, SIM2REASON demonstrates that simulator-generated supervision can help scale physical reasoning in language models while still requiring careful real-world validation as the framework expands beyond mechanics.

2.6 Appendix: Qualitative Examples

This appendix collects qualitative side-by-side comparisons between the base and RL-finetuned models on representative IPhO and JEE problems referenced in 2.4.

JEE Advanced 2017 Paper 2

QUESTION

A rocket is launched normal to the surface of the Earth, away from the Sun, along the line joining the Sun and the Earth.

Parameters:

- Mass of Sun = $300,000\times$ Mass of Earth
- Distance = $25,000\times$ Radius of Earth
- Escape Velocity (Earth) = 11.2 km/s

The minimum initial velocity (v_S) required for the rocket to be able to leave the Sun-Earth system is closest to:
[A] 22 km/s **[B]** 42 km/s **[C]** 62 km/s **[D]** 72 km/s

BASE MODEL

Step 1: Calculate Escape Velocity from Sun
Using the given ratios ($M_{\text{Sun}} = 3\times10^5 M_E$, $D = 2.5\times10^4 R_E$):

$v_{\text{sun}} = \sqrt{2GM_{\text{Sun}}/R}$
$v_{\text{sun}} \approx 54.78$ km/s

Step 2: Calculate Total Velocity
To escape both bodies, we sum the required velocities:

$v_{\text{total}} = v_{\text{earth}} + v_{\text{sun}}$
$v_{\text{total}} = 11.2 + 54.78$ km/s
ERROR: Linear addition
$v_{\text{total}} \approx 65.98$ km/s

Conclusion:
The calculated value (66 km/s) is closest to option [C].

Predicted Answer: [C] 62 km/s (Incorrect)

RL-FINETUNED MODEL

Step 1: Analyze the System
We need to escape the combined potential of Earth and Sun.
- Earth Escape Velocity: 11.2 km/s
- Sun Escape Velocity (at Earth's distance): ≈ 42 km/s
(based on standard solar data)

Step 2: Determine Effective Velocity
Since the Sun's gravitational well is significantly deeper than Earth's, the escape velocity is dominated by the Sun.
$v_{\text{total}} = \sqrt{v_{\text{sun}}^2 + v_{\text{earth}}^2}$

Correct: Energy Addition
Because $v_{\text{sun}}(42) \gg v_{\text{earth}}(11)$, the result is approximately v_{sun}.
$v_{\text{total}} \approx 42$ km/s

Conclusion:
The minimum velocity is dictated by the Sun's gravity.

Predicted Answer: [B] 42 km/s (Correct)

Figure 2.5: LLM answers before (left) and after (right) RL finetuning. Question adapted from JEE Advanced 2017 Paper 2.

IPhO 2005 Q1

QUESTION

A geosynchronous satellite of mass m is in a circular orbit of radius r_0 with velocity v_0.

An error causes the engine to fire, providing an instantaneous radial thrust Δv directed towards Earth. We characterize this boost by the parameter:

$$\beta = \frac{\Delta v}{v_0}$$

Task: Calculate the minimum boost parameter β_{esc} needed for the satellite to escape Earth's gravity.

BASE MODEL

Step 1: Orbital Velocity
$v_0 = \sqrt{GM_E/r_0}$

Step 2: Escape Condition
Escape velocity: $v_{\text{esc}} = \sqrt{2GM_E/r_0}$
Condition: $v_{\text{after}} \geq v_{\text{esc}}$

Since the thrust is directed radially inward, the effective velocity becomes reduced:
$v_{\text{after}} = v_0 - \Delta v$

ERROR: Linear subtraction. Fails to treat velocity as a vector (radial $\perp$ tangential).

Step 3: Solve for β
$\sqrt{GM/r} - \beta\sqrt{GM/r} \geq \sqrt{2GM/r}$
$(1 - \beta) \geq \sqrt{2}$
$1 - \sqrt{2} \geq \beta \implies \beta \approx -0.41$

(Model confuses signs and algebra)
$\beta \geq \sqrt{2} - 1 \approx 0.41$

Final Answer:

$\boxed{\beta_{\text{esc}} = \sqrt{2} - 1 \approx 0.41}$ (Incorrect)

RL-FINETUNED MODEL

Step 1: Energy Analysis
Initial Energy: $E_0 = -GMm/2r_0$
Escape Velocity needed: $v_{\text{esc}} = \sqrt{2GM/r_0}$

Step 2: Vector Addition
The kick is radial (Δv), perpendicular to the orbital velocity (v_0). The new velocity magnitude adds in quadrature:
$v_{\text{new}} = \sqrt{v_0^2 + (\Delta v)^2}$

Step 3: Escape Condition
$v_{\text{new}} \geq v_{\text{esc}}$
$\sqrt{v_0^2 + (\Delta v)^2} \geq \sqrt{2}v_0$
Substitute $\Delta v = \beta v_0$:
$\sqrt{v_0^2 + \beta^2 v_0^2} \geq \sqrt{2}v_0$
$\sqrt{1 + \beta^2} \geq \sqrt{2}$

Step 4: Solve for β
$1 + \beta^2 \geq 2$
$\beta^2 \geq 1 \implies \beta \geq 1$

Final Answer:

$\boxed{\beta_{\text{esc}} = 1}$ (Correct)

Figure 2.6: LLM answers before (left) and after (right) RL finetuning. Question adapted from IPhO 2005 Q1 "An Ill Fated Satellite".

JEE Advanced 2023 Paper 1

QUESTION

A small mass ($m = 0.10$ kg) moving with initial speed $u = 5.00$ m/s collides elastically with a bar (Mass $M = 1.00$ kg, Length $L = 0.20$ m) that is pivoted at one end. The collision occurs at a distance $L/2$ from the pivot.

Parameters:

- $m = 0.10$ kg; $u = 5.00$ m/s
- $M = 1.00$ kg; $L = 0.20$ m
- I_{bar} (about pivot) $= ML^2/3$

The velocity of the small mass (v) after the collision is closest to:
[A] 4.30 m/s **[B]** 5.00 m/s **[C]** -2.50 m/s **[D]** 2.15 m/s

BASE MODEL

Step 1: Conservation of Angular Momentum
$L_{\text{initial}} = m \cdot u \cdot (L/2)$
$L_{\text{initial}} = 0.1 \times 5.0 \times 0.1 = 0.005$
ERROR: Arithmetic (Should be 0.05)
$L_{\text{final}} = I_{\text{pivot}}\omega + mv(L/2)$
$0.005 = 0.00133\omega + 0.005v$

Step 2: Conservation of Kinetic Energy
Since collision is elastic, conserve energy.
$KE_{\text{initial}} = 0.5mu^2 = 1.25$ J
For the bar, we consider both rotation and translation:
$KE_{\text{bar}} = 0.5I_{\text{pivot}}\omega^2 + 0.5Mv_{\text{cm}}^2$
ERROR: Double counting energy
(Using I_{pivot} already accounts for motion)
of COM. Adding linear KE counts twice.)

Step 3: Solve System
The model attempts to fit the values... finds
$v \approx 4.30$ m/s, but associates it with wrong option due to internal confusion.

Predicted Answer: [B] 5.00 m/s (Incorrect)

RL-FINETUNED MODEL

Step 1: Conservation of Angular Momentum
Calculate initial angular momentum correctly:
$L_{\text{initial}} = m \cdot u \cdot (L/2)$
$L_{\text{initial}} = 0.10 \times 5.00 \times 0.10 = 0.05$
$L_{\text{final}} = I_{\text{pivot}}\omega + mv(L/2)$
$0.05 = (\frac{1}{3} \times 1.0 \times 0.2^2)\omega + 0.1v(0.1)$
$0.05 = 0.0133\omega + 0.01v$

Step 2: Conservation of Kinetic Energy
For a bar pivoting at the end, KE is purely rotational about pivot:
$KE_{\text{bar}} = 0.5I_{\text{pivot}}\omega^2$
Equation:
$1.25 = 0.05v^2 + 0.00665\omega^2$

Step 3: Solve System
From Momentum: $v = 5 - 1.33\omega$
Substitute into Energy...
Solving yields $\omega \approx 7.0$ rad/s.
Substituting ω back to find v:
$v = 5 - 1.33(7.0) \approx 4.30$ m/s

Conclusion:
Velocity decreases slightly to 4.30 m/s.

Predicted Answer: [A] 4.30 m/s (Correct)

Figure 2.7: LLM answers before (left) and after (right) RL finetuning. Question adapted from JEE Advanced 2023 Paper 1.

IPhO 2012 Q1

QUESTION

A ball is thrown with a fixed initial speed v_0 in a homogeneous gravitational field. The x-axis is horizontal, and the z-axis is vertical (opposing gravity g).

By adjusting the launching angle, the ball can hit any target within a region defined by the parabola: $z \leq z_0 - kx^2$

You can use this fact without proving it. Find the constants z_0 and k in terms of v_0 and g.

BASE MODEL

Step 1: Write Equations of Motion

The trajectory of a single projectile is given by:

$x = v_0 \cos(\theta) t$

$z = v_0 \sin(\theta) t - \frac{1}{2} g t^2$

Eliminating t, we get the trajectory equation:

$z = x \tan(\theta) - \frac{g x^2}{2 v_0^2 \cos^2(\theta)}$

Step 2: Match Coefficients

We compare this specific trajectory to the given boundary equation: $z = z_0 - kx^2$.

To find the constants, we assume the standard case for maximums, which is $\theta = 45°$.

- At 45 degrees, $\tan(\theta) = 1$.
- At 45 degrees, $\cos^2(\theta) = 0.5$.

Step 3: Solve for k

Looking at the x^2 term in the trajectory equation:

Coefficient $= \frac{g}{2 v_0^2 \cos^2(\theta)}$

Substitute $\cos^2(45) = 0.5$:

$k = \frac{g}{2 v_0^2 \cdot 0.5}$

$k = g / v_0^2$

ERROR: Assumed boundary is defined by the 45-degree path.

Conclusion:

$z_0 = v_0^2 / (2g)$

$k = g / v_0^2$

Predicted Answer: $z_0 = v_0^2/2g, k = g/v_0^2$ (Incorrect k)

RL-FINETUNED MODEL

Step 1: Analyze the Physical Meaning

The equation $z = z_0 - kx^2$ represents the "Envelope of Safety"-the outer boundary of all possible trajectories. We are not looking for one specific path, but the limits of all paths.

We can find the constants by identifying two critical edge cases on this boundary:

1. The maximum possible height ($z_{\max}$) where $x = 0$.
2. The maximum possible range ($x_{\max}$) where $z = 0$.

Step 2: Find z_0 (The Peak)

The maximum height is achieved by throwing the ball straight up ($\theta = 90°$).

Using energy conservation ($mgh = \frac{1}{2} m v^2$):

$z_0 = v_0^2 / (2g)$

Step 3: Find k (The Width)

The boundary touches the ground ($z = 0$) at the maximum possible range.

We know max range occurs at 45 degrees:

$x_{\max} = v_0^2 / g$

Now, substitute this point ($z = 0, x = v_0^2/g$) into the boundary equation:

$0 = z_0 - k x_{\max}^2$

$0 = \frac{v_0^2}{2g} - k (\frac{v_0^2}{g})^2$

Step 4: Solve for k

$k \frac{v_0^4}{g^2} = \frac{v_0^2}{2g}$

$k = \frac{v_0^2}{2g} \cdot \frac{g^2}{v_0^4}$

$k = \frac{g}{2 v_0^2}$

Conclusion:

$z_0 = v_0^2 / (2g)$

$k = g / (2 v_0^2)$

Predicted Answer: $z_0 = v_0^2/2g, k = g/2v_0^2$ (Correct)

Figure 2.8: LLM answers before (left) and after (right) RL finetuning. Question adapted from IPhO 2012 Question 1 "Focus on sketches".

IPhO 2018 Q1

QUESTION

Calculate the dimensionless coefficient ξ for the power $\mathcal{P}$ emitted in gravitational waves by a binary system.

Formula:

$$\mathcal{P} = \frac{G}{5c^5}\sum_{i,j}\left(\frac{\mathrm{d}^3 Q_{ij}}{\mathrm{d}t^3}\right)^2 = \xi\frac{G}{c^5}\mu^2 L^4 \Omega^6$$

Given Quadrupole Moments (where $k = 2\Omega$):
$Q_{ii} = \frac{\mu L^2}{2}(a_i + b_i \cos kt)$ and $Q_{ij} = \frac{\mu L^2}{2} c_{ij} \sin kt$ (for $i \neq j$)

Parameters:
$b_1 = 1, b_2 = -1, b_3 = 0$; $c_{12} = c_{21} = 1$ (others 0).

The value of ξ is: **[A]** 1.6 **[B]** 3.2 **[C]** 6.4 **[D]** 12.8

BASE MODEL

Step 1: Compute Third Time Derivatives
We differentiate the quadrupole moments three times.
$Q(t) \sim \frac{1}{2}\cos(2\Omega t)$

First deriv: $\sim \frac{1}{2}\cdot(2\Omega) = \Omega$
ERROR: Failed chain rule (missing factor of 2)
Second deriv: $\sim \Omega \cdot (\Omega) = \Omega^2$
Third deriv: $\sim \Omega^2 \cdot (2\Omega) = 2\Omega^3$

Resulting Coefficient: $A = 2$

Step 2: Square and Sum
We square the coefficient found in Step 1.
Square: $A^2 = 2^2 = 4$

Summing over the non-zero components (diagonal and off-diagonal contribute equally):
Sum $= 4$ (from diagonal) $+4$ (from off-diagonal) $= 8$

Step 3: Apply Formula
$\mathcal{P} = \frac{1}{5}\cdot$ Sum $\cdot$ (constants)
$\xi = \frac{8}{5} = 1.6$

Predicted Answer: [A] 1.6 (Incorrect)

RL-FINETUNED MODEL

Step 1: Compute Third Time Derivatives
We apply the chain rule consistently for the third derivative.
$Q(t) \sim \frac{1}{2}\cos(2\Omega t)$

First deriv: $\frac{1}{2}\cdot(-2\Omega)\sin(2\Omega t)$
Second deriv: $\frac{1}{2}\cdot(-4\Omega^2)\cos(2\Omega t)$
Third deriv: $\frac{1}{2}\cdot(8\Omega^3)\sin(2\Omega t)$

Resulting Coefficient: $A = 4$

Step 2: Square and Sum
We square the coefficient found in Step 1.
Square: $A^2 = 4^2 = 16$

Summing over the components (using $\sin^2 + \cos^2 = 1$ identity):
Sum $= 16$ (from diagonal) $+16$ (from off-diagonal) $= 32$

Step 3: Apply Formula
$\mathcal{P} = \frac{1}{5}\cdot$ Sum $\cdot$ (constants)
$\xi = \frac{32}{5} = 6.4$

Predicted Answer: [C] 6.4 (Correct)

Figure 2.9: LLM answers before (left) and after (right) RL finetuning. Question adapted from IPhO 2018 Question 1 "LIGO-GW150914".

IPhO 2013 Q1

QUESTION

A meteoroid from Comet Encke impacted Earth (the Maribo meteorite). Measurements show the meteorite formed shortly after the solar system's birth.

In its orbit around the Sun, the minimum and maximum distances between Comet Encke and the Sun are:

$$a_{\min} = 4.95 \times 10^{10} \text{ m} \quad \text{and} \quad a_{\max} = 6.16 \times 10^{11} \text{ m}$$

Calculate the orbital period t_{Encke} of Comet Encke. Express the final answer in SI units.

BASE MODEL

Step 1: Formula

Use Kepler's Third Law:

$T^2 = (4\pi^2/GM) \cdot a^3$

Step 2: Calculate Semi-major Axis

$a = (a_{\min} + a_{\max})/2$

$a = (4.95 \times 10^{10} + 6.16 \times 10^{11})/2$

$a = 33.275 \times 10^{10}$ m

Step 3: Calculate Period

Using $\mu = 1.327 \times 10^{20}$:

$T^2 = (33.275 \times 10^{10})^3 / 1.327 \times 10^{20}$

$T^2 = 3.7074 \times 10^{33} / 1.327 \times 10^{20}$

ERROR: Arithmetic. $a^3 \approx 3.7 \times 10^{34}$ (Order of magnitude off).

$T^2 \approx 2.794 \times 10^{13}$ s^2

$T \approx 5.285 \times 10^6$ s

$T \approx 61.37$ days

Final Answer:

$\boxed{61.37 \text{ days}}$ (Incorrect)

RL-FINETUNED MODEL

Step 1: Calculate Semi-major Axis

$a = (a_{\min} + a_{\max})/2$

$a = (4.95 \times 10^{10} + 61.6 \times 10^{10})/2$

$a = 3.3275 \times 10^{11}$ m

Step 2: Unit Conversion Strategy

Convert a to Astronomical Units (AU) to simplify Kepler's Law ($T^2 = a^3$ in years/AU).

1 AU $= 1.496 \times 10^{11}$ m

$a_{au} = 3.3275 \times 10^{11} / 1.496 \times 10^{11} \approx 2.225$ AU

Step 3: Apply Kepler's Law

$T^2 = (2.225)^3 \approx 10.96$

$T \approx \sqrt{10.96} \approx 3.31$ years

Step 4: Convert to SI

1 year $= 3.154 \times 10^7$ seconds

$T = 3.31 \times 3.154 \times 10^7$

$T \approx 1.04 \times 10^8$ seconds

Final Answer:

$\boxed{1.04 \times 10^8 \text{ s}}$ (Correct)

Figure 2.10: LLM answers before (left) and after (right) RL finetuning. Question adapted from IPhO 2013 Question 1 "The Maribo Meteorite".

Chapter 3

Iterative Refinement Improves Compositional Image Generation

3.1 Abstract

Text-to-image (T2I) models now produce highly realistic images, but they still have difficulty when a prompt combines multiple requirements at once, especially when several objects, attributes, and relationships must all be rendered correctly in the same scene. Methods that increase inference-time compute, such as running additional denoising steps or generating multiple candidates for later selection as in parallel sampling, can improve performance to some extent, yet these strategies often remain unreliable for strongly compositional prompts. A more effective approach is to treat image generation as a step-by-step refinement process, where an initial output from an image generator is repeatedly revised using feedback from a vision-language model critic that evaluates how well the image satisfies the prompt. This test-time procedure [16] is lightweight, does not depend on external priors or specialized auxiliary tools, and can be integrated with different image generators and critic models. Performance improvements are reported across several benchmarks, including gains of 16.9% on ConceptMix ($k = 7$), 13.8% on the 3D-Spatial portion of T2I-CompBench, and 12.5% on Visual Jenga scene decomposition relative to a compute-matched parallel sampling baseline. Human preference results also favor the iterative method, with selections of 58.7% compared with 41.3% for the baseline, indicating that progressive self-correction provides a broadly useful mechanism for generating images that better satisfy complex compositional prompts.

3.2 Introduction

Large language models (LLMs) have recently achieved major gains in reasoning performance, and an important part of this progress has come from increasing test-time compute [6, 31, 37]. One especially influential technique is chain-of-thought (CoT) prompting, in which the model is encouraged to reason step by step rather than produce an immediate answer [18, 37]. This simple change often leads to behaviors such as intermediate error checking, revision, and self-correction, which are closely tied to stronger performance on tasks that require multi-step reasoning. These developments suggest that modern generative systems can benefit not only from larger models or more data, but also from mechanisms that allow outputs to be improved through structured intermediate computation.

A key distinction between language and image generation lies in the form of supervision available during pre-training. LLMs are exposed to large amounts of text that often contain natural reasoning traces, including derivations, instructions, and logical explanations, so CoT prompting can build on patterns already present in the training data. Text-to-image (T2I) models, by contrast, are typically trained on image-caption pairs that describe visual content but do not contain explicit step-by-step reasoning. As a result, these models generally rely on one-shot generation and do not naturally acquire the ability to revise their outputs over multiple stages. This limitation becomes especially apparent for complex compositional prompts, where correct generation depends on simultaneously satisfying many object, attribute, and relation constraints.

An effective way to introduce self-correction into T2I generation is to replace single-pass synthesis with an iterative refinement pipeline. The proposed framework contains four components as shown in Figure 3.1: a T2I model image generator that generates an initial image, a vision-language model (VLM) critic that compares the image against the target prompt and proposes corrections, an image editor that applies these edits, and a verifier that measures final prompt alignment. This design differs from standard parallel sampling baselines [22, 47], where several images are generated independently and the best one is selected afterward. Parallel sampling increases diversity, but it does not allow the system to build on earlier partial successes. For highly compositional prompts involving many concept bindings, this is a serious limitation: if all constraints cannot be resolved within one forward pass, drawing more independent samples may still leave pass@k near zero. Sequential refinement instead allows only a subset of bindings to be handled at each stage, while preserving and improving components that were already generated correctly.

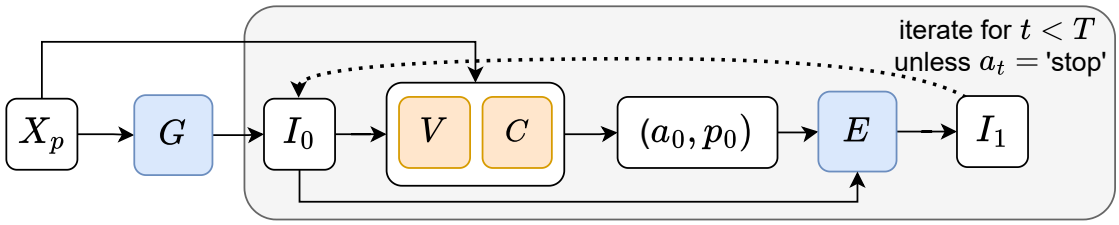

Figure 3.1: Overview of the iterative refinement framework with four components: a text-to-image generator, a vision-language model critic, an image editor, and a verifier.

This refinement view also distinguishes the approach from earlier sequential or tool-based systems such as GenArtist [35], CompAgent [36], and RPG [43], which often depend on specialized modules including layout generators, bounding-box tools, dragging operations, or object-removal pipelines. Such systems can become brittle because performance depends on multiple auxiliary components that may lag behind current foundation models or introduce compounding errors. With recent improvements in VLMs and image-editing models, a lighter and more general alternative becomes possible: strong critic feedback combined with standard editing is sufficient to obtain state-of-the-art compositional image generation without heavy task-specific engineering. Figure 3.2 illustrates this contrast qualitatively, showing that iterative refinement can solve a complex prompt under the same compute budget where parallel sampling fails even after four passes. Figure 3.3 reports corresponding quantitative gains, including a 16.9% higher all-correct rate on ConceptMix (concept binding = 7) [42] and a 13.8% improvement on the 3D-Spatial category of T2I-Bench [14] relative to compute-matched parallel sampling. Figure 3.5 further indicates an approximately $\sim 9+\%$ point advantage over methods such as GenArtist and RPG in highly compositional settings, and the same framework extends naturally to the Visual Jenga scene decomposition task discussed in Section 3.5 [3]. More broadly, these results support self-correction as a useful inductive principle for generative vision, suggesting that image models can benefit from iterative critique-and-revision processes in much the same way that CoT benefits reasoning in language models.

3.3 Related Work

Text-to-image (T2I) generation has advanced rapidly, but prompt fidelity remains difficult when a scene requires many interacting objects, attributes, and relations. Existing inference-time methods improve alignment in different ways, including classifier-free guidance [13], parallel sampling [7, 9], and grounding-based approaches that tie text more ex-

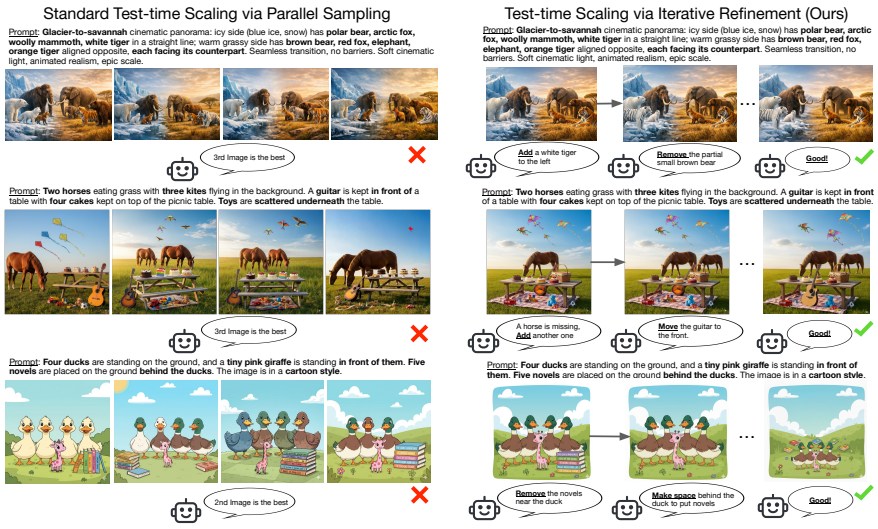

Figure 3.2: Qualitative comparison of iterative refinement versus parallel sampling under the same compute budget.

plicitly to image regions or objects [20, 21]. Iterative editing and refinement methods, such as SDEdit, InstructPix2Pix, and IterComp [5, 24, 48], further demonstrate that multi-step correction can improve prompt alignment over repeated generation stages. Related work on human-preference-guided evaluation and alignment also emphasizes the value of adaptive feedback during inference [17, 19, 41]. At the model and system level, modern T2I generators and compositional pipelines, including FLUX, DALL·E 3, GPT-Image-1.5, Qwen-Image, RPG, GenArtist, PARM, LLM Diffusion, and CompAgent, improve compositional generation through tool use, region-specific priors, or reinforcement-learning-based objectives [4, 21, 25, 26, 35, 36, 39, 43, 46]. In contrast, the method positioned here is training-free and relies on a simpler inference-time loop built from a vision-language model (VLM) critic, an image generator, and an image editor, rather than a large auxiliary tool stack.

A second line of related work comes from chain-of-thought (CoT) reasoning in large language models, where step-by-step prompting improves performance on difficult reasoning tasks [34, 37, 44]. Closely related self-refinement methods show that sequential feedback and revision can help models correct mistakes and solve problems more reliably over multiple stages [23]. This reasoning framework motivates an analogous view of compositional image generation: instead of requiring all constraints to be satisfied in a single forward pass, a critic can evaluate an intermediate image, identify missing or incorrect elements, and issue targeted refinement instructions for the next editing step. Under this interpretation, the VLM critic serves a role similar to CoT-style intermediate reasoning, allowing image synthesis to proceed

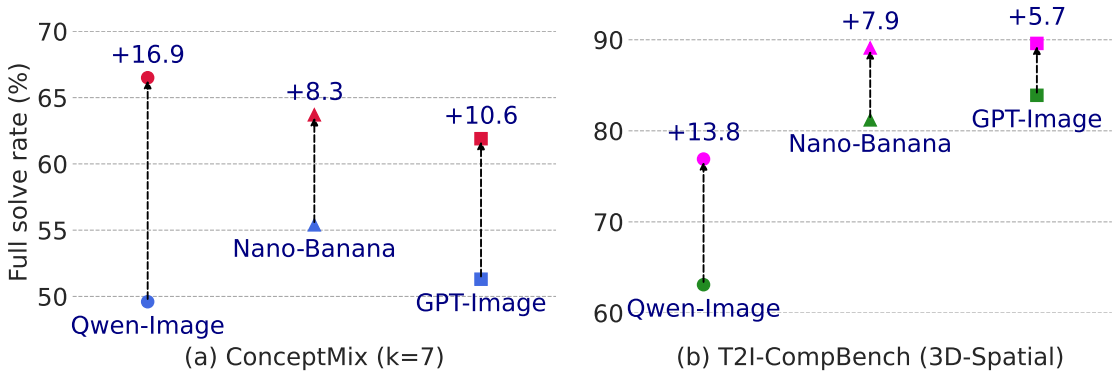

Figure 3.3: Quantitative improvements of iterative refinement over parallel sampling on ConceptMix and T2I-CompBench benchmarks.

through iterative assessment and correction rather than one-shot generation alone.

3.4 Method

Iterative refinement is used to generate an image I from a complex prompt P under a fixed inference-time budget. Instead of relying on a single text-to-image pass, the method treats generation as a sequence of guided updates distributed across multiple parallel streams. The framework contains four main components: a text-to-image generator G, an image-to-image editor E, a verifier V that scores alignment between a candidate image and the prompt, and a critic C that outputs both a refinement sub-prompt p_t and an action a_t. The total compute budget is written as $B = T \times M$, where T denotes the number of refinement rounds and M denotes the number of parallel streams. This parameterization exposes a depth–breadth trade-off: larger T permits deeper sequential correction, while larger M increases the number of candidate trajectories explored in parallel. As illustrated in Figure 3.1, this setup allows intermediate generations to be evaluated, critiqued, and selectively improved rather than discarded after a single forward pass.

At $t = 0$, each stream is initialized with an image $I_0^m \leftarrow G(P)$. At every subsequent round, the verifier computes an alignment score $s_t^m \leftarrow V(I_t^m, P)$, and the critic returns $(a_t^m, p_t^m) \leftarrow C(I_t^m, P)$. The critic chooses among four actions: **STOP**, which ends refinement for that stream; **BACKTRACK**, which reverts to the previous image I_{t-1}^m and edits it using the new sub-prompt; **RESTART**, which discards the current trajectory and regenerates from scratch conditioned on P and p_t^m; and **CONTINUE**, which directly edits the current image. These updates are expressed as $I_{t+1}^m \leftarrow E(I_{t-1}^m, p_t^m)$ for backtracking, $I_{t+1}^m \leftarrow G(P, p_t^m)$

Algorithm 1 Iterative Image Refinement over Parallel Streams with Critic Feedback

Require: Prompt P, generator G, image editor E, verifier V, critic C, parallel streams M, maximum rounds T

$B \leftarrow T \times M$
Initialize $\{I_0^m\}_{m=1}^M \leftarrow \{G(P)\}_{m=1}^M$
for $t = 1$ to T **do**
  for $m = 1$ to M **in parallel do**
    $s_t^m \leftarrow V(I_t^m, P)$
    $(a_t^m, p_t^m) \leftarrow C(I_t^m, P)$
    if $a_t^m =$ **STOP then**
      Mark stream m as complete
    else if $a_t^m =$ **BACKTRACK then**
      $I_{t+1}^m \leftarrow E(I_{t-1}^m, p_t^m)$
    else if $a_t^m =$ **RESTART then**
      $I_{t+1}^m \leftarrow G(P, p_t^m)$
    else if $a_t^m =$ **CONTINUE then**
      $I_{t+1}^m \leftarrow E(I_t^m, p_t^m)$
    end if
  end for
  $I_t^* \leftarrow \arg\max_m s_t^m$
  if all streams have emitted **STOP then**
    return I_t^*
  end if
end for
return I_T^*

for restarting, and $I_{t+1}^m \leftarrow E(I_t^m, p_t^m)$ for continuing. After each round, the best candidate is selected as $I_t^* = \arg\max_m s_t^m$, and the process stops either when all streams emit **STOP** or when the budget B is exhausted. In this way, difficult compositional prompts can be decomposed into a sequence of smaller corrections, making the overall procedure a visual analogue of chain-of-thought-style reasoning.

3.5 Experiments

Experimental evaluation is conducted across three recent text-to-image (T2I) model families: Qwen-Image [39], Gemini 2.5 Flash Image (Nano-Banana), and GPT-Image-1.5 [26]. The experiments measure compositional generation performance on ConceptMix [42], T2I-CompBench [14], and TIIF-Bench [38], and also extends the refinement framework to the Visual Jenga scene decomposition task [3]. These benchmarks cover complementary forms of difficulty: ConceptMix tests concept binding under increasing compositional complex-

ity from $k = 1$ to $k = 7$, T2I-CompBench evaluates attribute binding, spatial relations, numeracy, and open-world multi-object reasoning, and TIIF-Bench focuses on fine-grained instruction following, including 3D perspective, negation, text rendering, and 2D spatial control. Benchmark evaluation follows the original protocols, with a stronger multimodal language model serving as the final evaluator—Gemini-2.5-Pro or GPT-4o for ConceptMix and TIIF-Bench, and GPT-4V-style scoring for T2I-CompBench. Importantly, the in-loop critic and verifier are weaker models than the final evaluators; the primary experiments use Gemini-2.5-Flash inside the refinement loop.

A central part of the evaluation compares three inference-time strategies under matched compute: Parallel, Iterative (Iter), and Iterative+Parallel (Iter+Par). For Qwen-Image, the compute budget is set to $B = 16$ on ConceptMix and $B = 8$ on T2I-CompBench, while for GPT-Image-1.5 and Nano-Banana the experiments use $B = 12$ on ConceptMix and $B = 8$ on T2I-CompBench because of higher inference cost and closed-source access limits. Under the mixed strategy, compute is divided between a small number of parallel branches and the remaining iterative refinement steps; for Qwen-Image, this means two parallel branches and $B/2$ iterative updates. Table 3.1 shows that both iterative variants consistently outperform the budget-matched parallel-only baseline, especially as prompt complexity increases. On ConceptMix, Qwen Iter+Par improves the full solve rate over Qwen Parallel from 49.6% to 66.5% at $k = 7$, a gain of 16.9%, with similarly strong gains at $k = 5$ and $k = 6$ of 18.8% and 20.6%, respectively. Nano-Banana Iter+Par improves from 55.4% to 63.7% at $k = 7$ (+8.3%), and GPT-Image Iter+Par improves from 51.3% to 61.9% (+10.6%). These gains are not limited to the most difficult settings: measurable improvements also appear at lower binding counts such as $k = 1$, $k = 2$, and $k = 3$, indicating that iterative correction benefits both moderate and highly compositional prompts.

The same comparison reveals that iterative refinement is especially effective in categories requiring explicit spatial or quantitative control. Table 3.1 shows that for Qwen-Image, Iter+Par raises the Spatial score from 82.3 to 89.4, the 3D-Spatial score from 63.1 to 76.9, and the Numeracy score from 87.0 to 93.3, corresponding to gains of 7.1, 13.8, and 6.3, respectively. Nano-Banana also improves strongly in the same categories, with gains of 6.4 on Spatial, 7.9 on 3D-Spatial, and 9.8 on Numeracy, while GPT-Image gains 3.5, 5.7, and 4.6 in those categories. Figure 3.4 provides a finer-grained view of ConceptMix performance for Qwen-Image and shows that the largest category-level improvements occur in Spatial, Style, Shape, and Size, whereas Object and Color benefit less, likely because baseline performance in those categories is already relatively strong. These patterns indicate that iterative refinement is most useful when correct generation requires structured binding of relations,

geometry, or count-based constraints rather than simpler appearance attributes alone.

	ConceptMix full solve rate (%)							T2I-CompBench VLLM (GPT4o) score (1 to 100)							
Model	**k=1**	**k=2**	**k=3**	**k=4**	**k=5**	**k=6**	**k=7**	**Spatial**	**3DSpat**	**Numer**	**Shape**	**Color**	**Texture**	**Non-Spat**	**Cmplex**
Qwen Parallel	92.8	82.5	74.3	69.2	60.1	51.2	49.6	82.3	63.1	87.0	87.2	92.6	**96.2**	92.8	93.4
Qwen Iter (ours)	96.1	91.4	87.0	82.1	**79.6**	67.4	64.3	87.4	**77.3**	91.1	91.2	92.4	95.1	**94.8**	94.8
Qwen Iter.+Par. (ours)	**96.5**	**91.7**	**87.4**	**82.2**	78.9	**71.8**	**66.5**	**89.4**	76.9	**93.3**	**90.1**	92.6	95.8	94.7	**95.0**
	+3.7	+9.2	+13.1	+13.0	+18.8	+20.6	+16.9	+7.1	+13.8	+6.3	+2.9	+0.0	-0.4	+1.9	+1.6
Nano-Banana Parallel	93.8	88.8	86.6	78.4	65.8	61.7	55.4	84.7	81.2	84.3	88.5	89.8	95.0	**96.8**	91.0
Nano-Banana Iter (ours)	94.1	90.4	87.2	81.3	**73.5**	64.6	63.6	90.6	87.8	93.9	89.9	89.7	**95.1**	95.8	**94.7**
Nano-Banana Iter.+Par. (ours)	93.8	**91.0**	**87.5**	**82.8**	71.4	**69.8**	**63.7**	**91.1**	**89.1**	**94.1**	**88.8**	**92.1**	94.8	96.7	94.5
	+0.0	+2.2	+0.9	+4.4	+5.6	+8.1	+8.3	+6.4	+7.9	+9.8	+0.3	+2.3	-0.2	-0.1	+3.5
GPT-Image Parallel	94.2	89.2	88.1	76.7	71.0	69.5	51.3	87.5	83.9	88.6	88.5	91.6	**92.5**	95.3	92.9
GPT-Image Iterative (ours)	96.0	91.4	90.6	**85.4**	72.0	69.6	58.9	89.6	**90.0**	92.7	**92.1**	**91.9**	92.0	**95.5**	93.0
GPT-Image Iter.+Par. (ours)	**97.7**	**94.2**	**91.1**	84.6	**79.5**	**76.8**	**61.9**	**91.0**	89.6	**93.2**	90.9	91.1	92.3	95.3	**93.1**
	+3.5	+5.0	+3.0	+7.9	+8.5	+7.3	+10.6	+3.5	+5.7	+4.6	+2.4	-0.5	-0.2	+0.0	+0.2

Table 3.1: Performance comparison of parallel sampling, iterative refinement, and combined iterative-parallel sampling across three state-of-the-art T2I models on ConceptMix and T2I-CompBench.

A broader benchmark comparison shows that the refinement framework remains effective beyond the two main compositional datasets and also compares favorably with prior compositional pipelines. Figure 3.5 compares the approach with GenArtist [35], RPG [43], and IterComp [48]. While these baselines remain competitive at lower concept counts, the gap widens as the number of concepts increases, suggesting that simple critic-guided refinement scales more effectively than pipelines that depend on layout planners, region priors, object detectors, or other specialized tools. Table 3.2 reports results on TIIF-Bench, where Qwen-Iter+Par reaches an overall score of 87.4, compared with 85.2 for Qwen-Parallel. The gains are especially visible in reasoning-heavy settings: Basic Following improves from 85.2 to 88.1, with Reasoning improving from 77.7 to 85.4; in Advanced Following, Relation+Reasoning improves from 77.8 to 80.5, and Text rendering improves from 93.7 to 97.7. These results show gains of about 7.7 on basic reasoning prompts, 2.7 on advanced Relation+Reasoning, and roughly 4.0 on text rendering relative to the Qwen parallel baseline. These results indicate that the refinement loop is not limited to a single benchmark family, but transfers to broader instruction-following scenarios that require combinations of attribute control, relational understanding, and symbolic rendering.

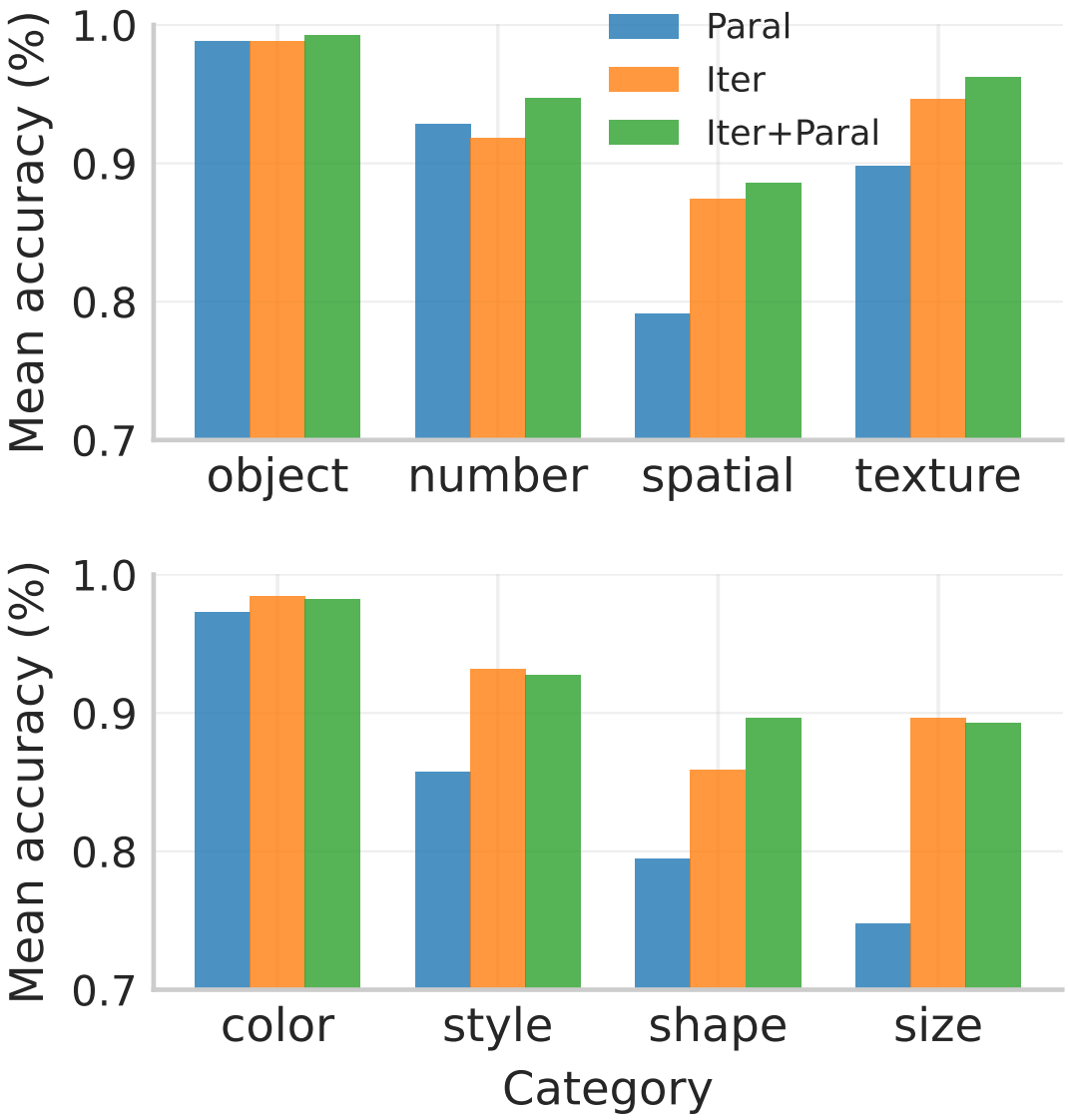

Figure 3.4: Category-level comparison of iterative refinement versus parallel sampling on ConceptMix for Qwen-Image, showing that the largest gains occur in Spatial, Style, Shape, and Size categories.

Model	Overall	Basic Following				Advanced Following						Designer
		Avg	Attribute	Relation	Reasoning	Avg	Attr+Rel	Attr+Reas	Rel+Reas	Style	Text	Real World
FLUX.1 [dev] [4]	71.1	83.1	87.1	87.3	75.0	65.8	67.1	73.8	69.1	66.7	43.8	70.7
SD 3	67.5	78.3	83.3	82.1	71.1	61.5	61.1	68.8	51.0	66.7	59.8	63.2
Janus-Pro-7B	66.5	79.3	79.3	78.3	80.3	59.7	66.1	70.5	67.2	60.0	28.8	65.8
MidJourney v7	68.7	77.4	77.6	82.1	72.6	64.7	67.2	81.2	60.7	83.3	24.8	68.8
Seedream 3.0	86.0	87.1	90.5	**89.9**	80.9	79.2	79.8	77.2	75.6	**100.0**	97.2	83.2
Qwen-Parallel [39]	85.2	85.2	89.7	88.3	77.7	80.6	**81.9**	79.6	77.8	89.7	93.7	90.4
Qwen-Iter	85.4	85.0	**92.0**	80.5	82.3	81.3	80.8	80.1	80.2	86.2	97.6	88.4
Qwen-Iter+Par	**87.4**	**88.1**	90.5	88.1	**85.4**	**81.5**	81.4	**82.0**	**80.5**	90.0	**97.7**	**92.0**

Table 3.2: Performance comparison across prominent open-source text-to-image models on TIIF-Bench.

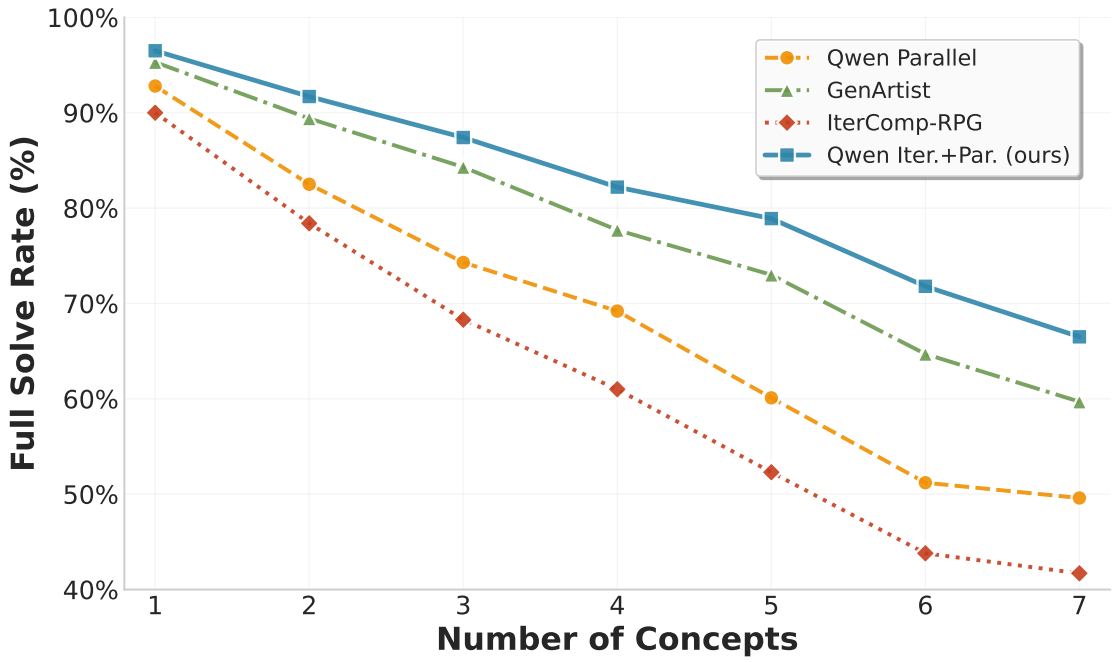

Figure 3.5: Comparison of iterative refinement against GenArtist, RPG, and IterComp on ConceptMix across increasing concept counts.

Qualitative analysis and human evaluation provide additional evidence that the gains come from meaningful intermediate corrections rather than only better final-image selection. Figure 3.6 traces multi-step refinement trajectories for prompts such as *"a mouse hidden by a key"*, *"a tiny red carrot positioned inside a huge metallic bee, depicted in a cubist style"*, and *"a lively pink flamingo spreading its wings as it dances in a rainforest"*. These examples illustrate the role of the critic action space: **Continue** is used to make targeted local improvements, **Restart** is used when the current image is too far from the desired composition or style, and **Backtrack** is used when a later edit introduces a worse interpretation than

an earlier state. Human evaluation is performed on 150 randomly sampled prompts from ConceptMix and T2I-CompBench, with three raters per prompt. Figure 3.7 reports that the iterative method is preferred 58.7% of the time, compared with 41.3% for the parallel-only baseline. Inter-annotator agreement is 85.3%, and agreement between human judgments and the language-model evaluator is 83.4%, which supports the use of the multimodal evaluator as a reasonably reliable automatic scoring mechanism.

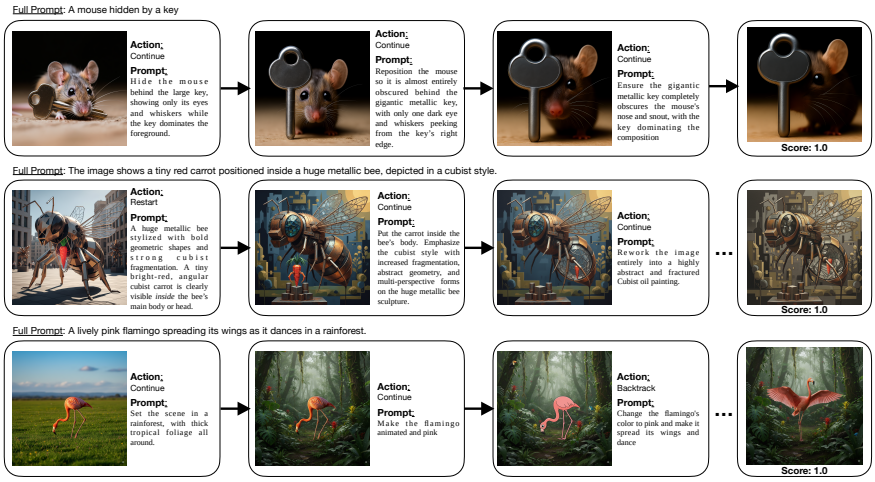

Figure 3.6: Multi-step refinement trajectories illustrating the roles of the Continue, Restart, and Backtrack critic actions across several compositional prompts.

Figure 3.7: Human preference evaluation comparing iterative refinement against parallel sampling on 150 randomly sampled prompts.

The refinement framework is also extended to Visual Jenga, where the task shifts from generation to progressive scene decomposition under physical plausibility constraints. Starting from a cluttered image, the system must identify an object to remove, generate an editing phrase such as *"remove the red mug from the table"*, and produce the next scene while keeping all remaining content coherent. Here, the VLM acts first as a proposer that suggests

the next removal step and then as a critic that checks whether the specified object was removed cleanly and whether the new image avoids artifacts, hallucinations, identity drift, or implausible scene changes. A step is retried with revised feedback until the critic accepts it or the per-step compute budget is exhausted. Using GPT-Image-1.5, the iterative system is compared with a compute-matched parallel baseline that generates four candidates per removal step. On the full decomposition subset of 56 scenes, Table 3.3 shows that the full solve rate increases from 64.29% for GPT-Parallel to 76.79% for GPT-Iter, indicating that iterative refinement significantly improves full solve rate over compute-matched parallel sampling. Figure 3.8 illustrates how the critic detects specific failures, such as residual shadows after ladder removal, incorrect book-count updates, or unintended identity changes in the background, and then uses these observations to produce more precise correction prompts in later iterations.

	GPT-Parallel	**GPT-Iter (ours)**
Full solve rate (%) (↑)	64.29	76.79

Table 3.3: Results on Visual Jenga full scene decomposition.

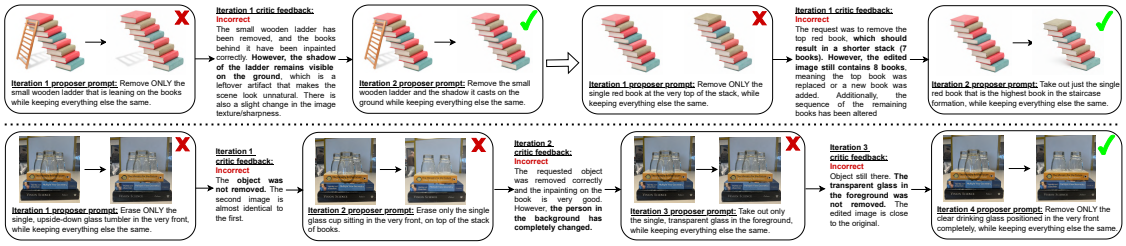

Figure 3.8: Visual Jenga scene decomposition examples showing how the critic detects failures such as residual shadows, incorrect object counts, and identity drift, and uses targeted feedback to guide subsequent editing iterations.

Ablation studies clarify how performance depends on compute allocation, critic quality, and action-space design. Table 3.4 studies the budget decomposition $B = I \times P$, where I is the number of iterative steps and P is the number of parallel branches. Across budgets $B \in \{1, 2, 4, 8, 16\}$, larger iterative allocations generally outperform larger parallel allocations once the total budget is at least moderate. For example, at $B = 4$, the purely iterative setting $(I = 4, P = 1)$ achieves 48.4 on ConceptMix and 86.4 on T2I-Avg, compared with 41.1 and 84.9 for the purely parallel setting $(I = 1, P = 4)$. At $B = 8$, the fully iterative setting $(8, 1)$ reaches 60.0 on ConceptMix and 89.9 on T2I-Avg, while the fully parallel setting $(1, 8)$ achieves only 44.8 and 86.5. The best result at $B = 16$ comes from a mixed but

iteration-heavy allocation, $(I = 8, P = 2)$, which reaches 69.6 on ConceptMix and 92.6 on T2I-Avg, slightly exceeding the fully iterative $(16, 1)$ setting. Figure 3.9 supports the same conclusion visually: high-iteration regimes consistently outperform low-iteration regimes at equal total compute, suggesting that a primarily iterative strategy with a small amount of parallel exploration is the most effective operating point.

I	P	$I \times P$	CMix(k$\{5, 6, 7\}$)	T2I-Avg
1	1	1	32.3	79.8
1	2	2	35.6	82.3
2	1	2	**37.2**	**82.6**
1	4	4	41.1	84.9
2	2	4	41.3	84.7
4	1	4	**48.4**	86.4
1	8	8	44.8	86.5
2	4	8	43.9	87.4
4	2	8	57.6	**90.2**
8	1	8	**60.0**	89.9
1	16	16	52.1	87.9
2	8	16	53.4	89.0
4	4	16	66.3	91.7
8	2	16	**69.6**	**92.6**
16	1	16	69.2	92.1

Table 3.4: Average Accuracy of configurations sorted by total compute; I = iterative steps, P = parallel steps.

Further ablations show that both the critic model and the critic action space materially affect performance. Table 3.5 compares several critic backbones on a ConceptMix subset: Gemini-2.5-Flash yields a solve rate of 69.7%, Gemini-Pro improves this to 74.0%, GPT-5 reaches 72.3%, and the smaller open-source Qwen3-VL-32B drops to 66.3%. This indicates that refinement quality depends strongly on the reasoning ability of the critic itself. Table 3.6 then evaluates ablations of the action space. The full action space achieves 69.7%, while removing **Backtrack** reduces performance to 68.0%, removing **Fresh Start** reduces it to 67.7%, and removing both reduces it further to 67.3%. The overall experimental picture is therefore consistent across quantitative benchmarks, human judgments, scene-decomposition tasks, and ablations: iterative self-correction provides a stronger use of inference-time compute than parallel sampling alone, and the most effective configuration combines a capable VLM critic with an iteration-dominant search strategy and a flexible action space that supports continuation, restart, and recovery from failed edits.

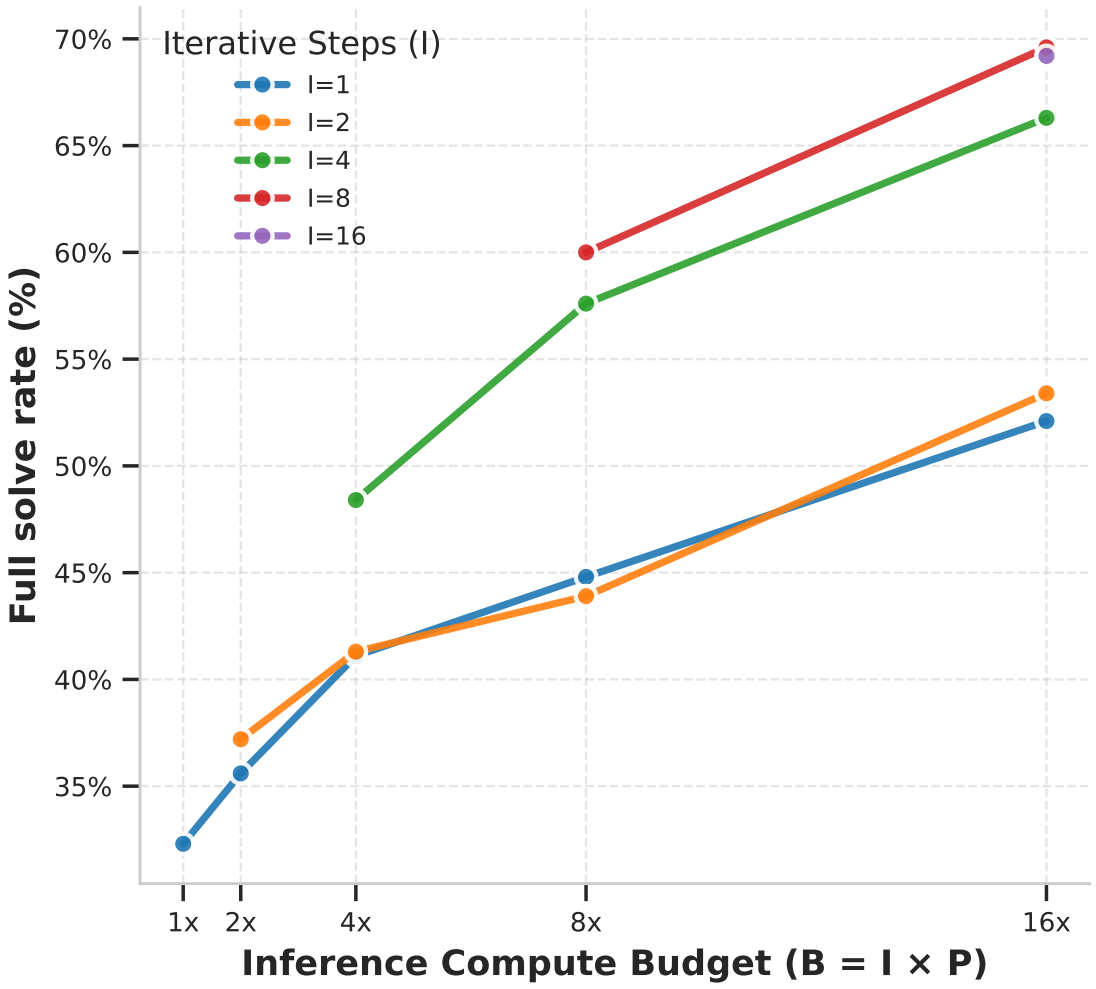

Figure 3.9: ConceptMix performance as a function of compute budget allocation between iterative and parallel strategies, showing that iteration-dominant regimes consistently outperform parallel-dominant ones.

Critic VLM	Solve rate (%)
Gemini-2.5-Flash	69.7
Gemini-Pro	74.0
GPT-5	72.3
Qwen3-VL-32B	66.3

Table 3.5: Impact of choice of critic VLM on performance.

Action Space	Solve rate (%)
Full action space	69.7
w/o Backtrack	68.0
w/o Fresh Start	67.7
w/o Backtrack & Fresh Start	67.3

Table 3.6: Impact of action space components on performance.

3.6 Conclusion

Iterative refinement serves as a simple and broadly applicable inference-time strategy for improving the compositional image generation ability of text-to-image (T2I) models by placing a vision-language model (VLM) critic in a refinement loop with an image generator and editor. Rather than relying on parallel sampling alone, the framework progressively revises intermediate outputs, which leads to stronger performance across major T2I model families, including Nano-Banana, Qwen-Image, and GPT-Image-1.5. The method achieves state-of-the-art results on compositional generation benchmarks such as ConceptMix, T2I-CompBench, and TIIF-Bench, and it also transfers effectively to the Visual Jenga scene decomposition task, indicating that the same refinement principle applies beyond standard prompt-to-image synthesis. Qualitative analyses further clarify how critic-guided edits progressively correct compositional errors, while human evaluation confirms that these improvements are meaningful beyond automatic benchmark scores. Additional ablations on compute allocation, critic model choice, and critic action space show that performance depends not only on total inference budget but also on how that budget is distributed between iterative and parallel search, as well as on the quality and flexibility of the critic itself. Overall, the framework demonstrates that structured self-correction at inference time provides a practical and effective alternative to conventional scaling strategies for compositional image generation.

Chapter 4

Conclusion

In aggregate, this work shows that explicit feedback can improve foundation models through both post-training and inference-time refinement. In scientific reasoning, the SIM2REASON framework demonstrates that physics simulators can serve as a scalable source of supervision by procedurally generating diverse scenes, converting simulation traces into verifiable question-answer pairs, and using RLVR to produce consistent zero-shot gains on real-world physics benchmarks even without natural QA data. In compositional image generation, iterative refinement shows that placing a VLM critic inside a generator-editor loop enables progressive correction of intermediate outputs, yielding strong gains across major text-to-image model families and state-of-the-art performance on several compositional benchmarks, with benefits that extend beyond standard prompt-to-image synthesis. Despite operating in different domains, both approaches point to the same broader conclusion: meaningful progress need not come only from larger models, larger datasets, or more compute, but can also come from feedback mechanisms that support structured self-correction and refinement. The shared message across these domains is that reasoning improves when models are given structured opportunities to check and correct their behavior, whether those checks come from physical simulation during training or visual critique during inference. By emphasizing scalable supervision and iterative refinement, this work points toward reasoning systems that are more capable, robust, and reliable across scientific and generative tasks.

Bibliography

[1] AIME. AIME problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2025. Accessed: 2026-01-29.

[2] Daman Arora, Himanshu Gaurav Singh, and Mausam. Have LLMs advanced enough? a challenging problem solving benchmark for large language models. *arXiv preprint arXiv:2305.15074*, 2023.

[3] Anand Bhattad, Konpat Preechakul, and Alexei A. Efros. Visual jenga: Discovering object dependencies via counterfactual inpainting. *arXiv preprint arXiv:2503.21770*, 2025.

[4] BlackForest Labs. FLUX. https://github.com/black-forest-labs/flux, 2024.

[5] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[6] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[7] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–10, 2023.

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

[9] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023.

[10] Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for LLM evaluation. https://github.com/huggingface/lighteval, 2023.

[11] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[14] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023.

[15] International Physics Olympiad. IPhO problems and solutions archive. https://www.ipho-new.org/statutes-syllabus, 2025. Problems spanning 1967–2025. Accessed: 2026-01-29.

[16] Shantanu Jaiswal, Mihir Prabhudesai, Nikash Bhardwaj, Zheyang Qin, Amir Zadeh, Chuan Li, Katerina Fragkiadaki, and Deepak Pathak. Iterative refinement improves compositional image generation. *arXiv preprint arXiv:2601.15286*, 2026.

[17] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

[19] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

[20] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521, 2023.

[21] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. LLM-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *arXiv preprint arXiv:2305.13655*, 2023.

[22] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[23] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[24] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

[25] OpenAI. DALL·E 3. https://openai.com/research/dall-e-3, 2023.

[26] OpenAI. GPT-image-1. https://openai.com/index/introducing-4o-image-generation, 2025.

[27] Mihir Prabhudesai, Aryan Satpathy, Yangmin Li, Zheyang Qin, Nikash Bhardwaj, Amir Zadeh, Chuan Li, Katerina Fragkiadaki, and Deepak Pathak. Solving physics olympiad via reinforcement learning on physics simulators. *arXiv preprint arXiv:2604.11805*, 2026.

[28] Gabriel Sarch, Snehesh Saha, Nishanth Khandelwal, Aayush Jain, Michael J. Tarr, Aviral Kumar, and Katerina Fragkiadaki. Grounded reinforcement learning for visual reasoning. *arXiv preprint arXiv:2505.23678*, 2025.

[29] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[30] Itamar Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

[31] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[32] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[33] H. C. Verma. *Concepts of Physics: Part 1*. Concepts of Physics. Bharati Bhawan Publishers & Distributors, Patna, India, 2017. ISBN 9788177091878.

[34] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[35] Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. GenArtist: Multimodal LLM as an agent for unified image generation and editing. *Advances in Neural Information Processing Systems*, 37:128374–128395, 2024.

[36] Zhenyu Wang, Enze Xie, Aoxue Li, Zhongdao Wang, Xihui Liu, and Zhenguo Li. Divide and conquer: Language models can plan and self-correct for compositional text-to-image generation. *arXiv preprint arXiv:2401.15688*, 2024.

[37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[38] Xinyu Wei, Jinrui Zhang, Zeqing Wang, Hongyang Wei, Zhen Guo, and Lei Zhang. TIIF-Bench: How does your T2I model follow your instructions? *arXiv preprint arXiv:2506.02161*, 2025.

[39] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Shengming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[40] Feng Wu, Wei Xuan, Hanyang Qi, Xin Lu, Ang Tu, Li Erran Li, and Yejin Choi. DeepSearch: Overcome the bottleneck of reinforcement learning with verifiable rewards via Monte Carlo tree search. *arXiv preprint arXiv:2509.25454*, 2025.

[41] Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2096–2105, 2023.

[42] Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. ConceptMix: A compositional image generation benchmark with controllable difficulty. *Advances in Neural Information Processing Systems*, 37:86004–86047, 2024.

[43] Ling Yang, Zhaochen Yu, Chenlin Meng, Minkai Xu, Stefano Ermon, and Bin Cui. Mastering text-to-image diffusion: Recaptioning, planning, and generating with multimodal LLMs. In *Forty-First International Conference on Machine Learning*, 2024.

[44] 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. *Advances in Neural Information Processing Systems*, 36:11809–11822, 2023.

[45] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Wenhao Dai, Tiantian Fan, Gaohang Liu, Lin Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxin Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiezhong Chen, Chenggang Wang, Hao Yu, Yufeng Song, Xingyan Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yu Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[46] Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Ziyu Guo, Haoquan Zhang, Manyuan Zhang, Jiaming Liu, Peng Gao, and Hongsheng Li. Let's verify and reinforce image generation step by step. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28662–28672, 2025.

[47] Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. *arXiv preprint arXiv:2505.23614*, 2025.

[48] Xinchen Zhang, Ling Yang, Guohao Li, Yaqi Cai, Jiake Xie, Yong Tang, Yujiu Yang, Mengdi Wang, and Bin Cui. IterComp: Iterative composition-aware feedback learning from model gallery for text-to-image generation. *arXiv preprint arXiv:2410.07171*, 2024.

[49] Chunyang Zheng, Shunian Liu, Mouxiang Li, Xiao-Hu Chen, Bowen Yu, Chang Gao, Kai Dang, Yanfei Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[50] Siyuan Zheng, Qian Cheng, Jiazheng Yao, Moye Wu, Hongyi He, Ning Ding, Yuxuan Cheng, Shengding Hu, Luoyi Bai, Dian Zhou, Ganqu Cui, and Peng Ye. Scaling physical reasoning with the physics dataset. *arXiv preprint arXiv:2506.00022*, 2025.