REFINING PROBLEM SPECIFICATIONS VIA TRACE-LEVEL DIAGNOSIS FOR REASONING DATA SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic data has become central to LLM post-training, but existing synthesis methods differ in what they tell the generator to synthesize. We call this conditioning signal the *specification*. Prior methods often use task descriptions or capability labels derived from problems the student answered incorrectly. These signals identify broad weak regions, but they do not explain where the student's reasoning failed. We propose Diagnosis-Driven Synthesis (DDS), a framework that uses the student's incorrect solution trace as a more specific synthesis signal. DDS first performs trace-centric diagnosis: a teacher inspects each failed solution and produces an actionable directive describing where the reasoning broke and what a new problem should require. DDS then introduces *diagnostic crossover*, which pairs two such diagnoses and asks a generator to create a single coherent problem that requires both weaknesses to be repaired jointly. A verifier filters generated candidates for well-posedness and solution correctness before fine-tuning. On math reasoning, DDS recovers $+12.6$ and $+8.0$ Pass@1 points on the student's failed problems over the base student on Qwen2.5-3B-Instruct and Llama-3.2-1B-Instruct, and improves the held-out math average by $+7.54$ and $+8.32$ points. At the same patch budget, DDS exceeds the strongest baseline by $+2.82$ and $+3.18$ points on the held-out math average. Iterative re-diagnosis further lifts the math average to 48.1. Beyond math, DDS reaches 59.93 on a seven-benchmark multimodal average, $+2.26$ over the strongest baseline.

1 INTRODUCTION

Modern post-training increasingly runs on synthetic data. As strong generators become widely available, the main bottleneck is shifting from how much data we can collect to what data we should synthesize. Existing work answers this question through different synthesis recipes. Some methods generate problems based on task descriptions (Wang et al., 2025; Kaur et al., 2025); others use model failures to guide generation (Zeng et al., 2025; Pan et al., 2025; An et al., 2024). These recipes have each shown value, but they are usually presented as independent synthesis heuristics rather than as instances of a shared principle. As a result, a unified account of what makes one recipe more effective than another for a given student model, or how to systematically improve synthesis methods, still remains unsolved.

We argue that these recipes differ along a hidden axis: the **specification** that conditions the generation of new synthetic data. Viewed through this lens, existing methods can be understood as making the synthesis specification progressively more informative (Figure 1). At the coarsest level, the specification is student-agnostic: generation is conditioned only on a task or benchmark description, and the generator is asked to populate different regions of the task space (Wang et al., 2025; Kaur et al., 2025). Such specifications can improve diversity and coverage, but they provide little information about which examples would repair the failure modes of the particular student being trained.

Recent failure-aware methods add student-specific information by first identifying the problems that the student answers incorrectly. Capability-profiling methods such as EvalTree (Zeng et al., 2025) use these failed problems to build or populate a hierarchical capability tree: failed problems are grouped under capability nodes, and nodes with poor student performance are treated as weak capabilities. The generator is then asked to write new problems that test those weak capability labels, such as "analyzing geometric relationships using trigonometric principles." This is more student-specific than generating

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

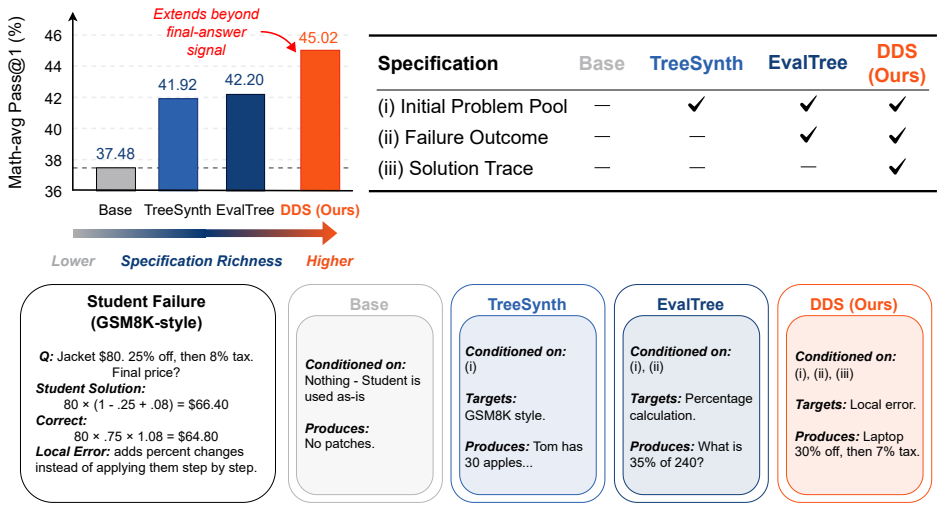


Figure 1: **Specification richness as a lens for reasoning data synthesis.** For an illustrative GSM8K-style failure, TreeSynth sees only the initial problem pool and writes another GSM8K-style problem. EvalTree also sees the failure outcome and targets its capability label (percentage calculation), which names the skill but not the error. DDS also reads the solution trace and targets the step that failed, applying the discount and the tax in sequence; the DDS bar includes diagnostic crossover.

from a task description alone, because the synthesis is guided by the student's observed failures. However, the failure signal is still outcome-level: it tells us which problems the student got wrong and which broad capability they belong to, but not where the student's reasoning broke inside each solution. As a result, many different trace-level mistakes can collapse into the same capability label. For example, two geometry failures may both fall under the same trigonometric-reasoning capability, even though one error comes from applying right-triangle trigonometry without establishing a right triangle and another comes from mishandling an obtuse-angle identity, $\sin(180^\circ - \theta) = \sin(\theta)$. DDS instead uses a richer specification: it diagnoses the student's incorrect solution trace and conditions synthesis on the specific reasoning error to be repaired.

This specification view suggests a systematic way to make synthesis more targeted to improve data synthesis: use the student's failed solution trace, rather than only the final-answer outcome or a broad capability label, as the basis for generation. Toward that end, we introduce Diagnosis-Driven Synthesis (DDS), a framework that turns failed traces into specifications for synthetic data generation. We refer to the initial pool of problems used to elicit student errors as the seed set, and to the student's incorrect attempts on those problems as seed failures. DDS first performs trace-centric diagnosis: a teacher inspects the student's incorrect solution and creates an actionable directive identifying where the reasoning went wrong and what constraint, inference, or concept a new problem should require. Because this directive is trace-specific, two failures that share the same outcome-level failure signal or capability label can still yield different synthesis specifications.

However, reasoning failures are not always isolated. A student may handle two skills separately but fail when a problem requires them to be coordinated. DDS therefore introduces diagnostic crossover, which uses a pair of trace-level diagnoses as the specification for synthesizing a single coherent problem. The resulting problem requires the student to repair both diagnosed weaknesses jointly, rather than practicing each weakness in isolation. Finally, a verifier filters generated candidates for well-posedness and solution correctness before supervised fine-tuning (SFT), so the added specificity of the specification does not come at the cost of lower data quality.

Our main contribution is to make the specification an explicit design choice in reasoning data synthesis and to enrich it in two ways: trace-centric diagnosis reveals where the reasoning broke, and diagnostic crossover composes two diagnosed failures into one problem. Empirically, DDS outperforms coarse-specification synthesis methods. On Qwen2.5-3B-Instruct, DDS recovers $+12.6$ Pass@1 points on the student's seed failures and improves the held-out math average by $+7.54$ points; on Llama-3.2-1B-Instruct, the corresponding gains are $+8.0$ and $+8.32$. The same framework

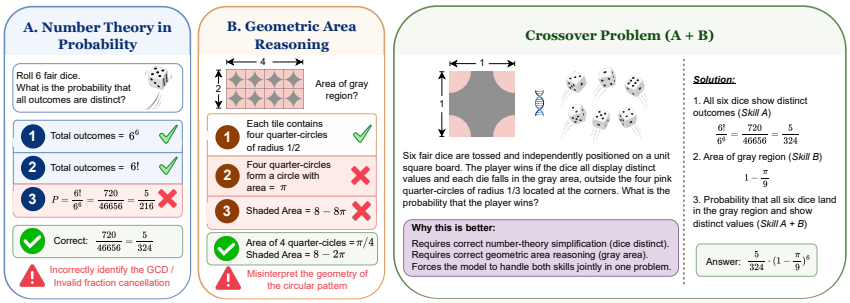


Figure 2: **Worked example of DDS.** The student fails two seed problems for different reasons: an invalid simplification in a probability calculation, and a misreading of the geometry of a shaded region. The teacher emits a trace-level diagnosis for each failure, which becomes part of the synthesis specification. Diagnostic crossover then composes the two diagnoses into a single new problem whose solution requires repairing both weaknesses.

extends to vision-language post-training and reaches an average of 59.93 across seven multimodal benchmarks.

2 Related Work

Specifications for synthetic data generation. Recent work on reasoning data synthesis differs less in whether it uses a strong generator than in what information is given to that generator. We call this conditioning information the *specification*. Student-agnostic synthesis methods condition on task descriptions, seed pools, or skills extracted from task-level sources, and use these signals to broaden or diversify the generated problem space (Wang et al., 2025; Kaur et al., 2025). These specifications can improve coverage, but they do not decide what to generate from the reasoning error made by a particular student. Failure-aware synthesis adds student information, but often at a coarse granularity. EvalTree (Zeng et al., 2025), for example, summarizes performance into weak capability nodes and asks the generator to test those capabilities. This gives a stronger signal than a task description, but many different reasoning errors can still collapse into the same capability label. DDS uses a different specification: the student's failed solution trace, distilled into a diagnosis of where the reasoning broke and what a new problem should require. **Failure-driven training and trace-level feedback.** A complementary line uses failures directly for training rather than as conditions for synthesizing new problems. Mistake-correction methods such as LeMa and LEMMA pair incorrect traces with teacher-corrected solutions and fine-tune the student on corrected supervision (An et al., 2024; Pan et al., 2025). These methods show that mistakes are useful supervision, but the training signal remains anchored to the original or surface-augmented questions. DDS instead uses the diagnosed failure as a condition for generating new problems, and diagnostic crossover composes two diagnosed failures into one problem requiring joint repair. More broadly, DDS follows the view that natural-language traces can provide richer learning signals than final outcomes alone; we defer a broader discussion of reflective optimization, weakness profiling, and additional synthesis methods to Appendix A.

3 Diagnosis-Driven Synthesis

We now instantiate the specification view from Section 1 as a concrete data-synthesis framework. Section 3.1 defines the seed failures, the synthesis specification, and the patching objective. Section 3.2 then describes DDS in three stages: trace-centric diagnosis, patch synthesis with diagnostic crossover, and quality control. The accepted patches are used for standard supervised fine-tuning, and the same procedure can be repeated after the student is updated.

3.1 Problem Setup

Let $\mathcal{Q}$, $\mathcal{R}$, and $\mathcal{A}$ denote the question space, the solution-trace space, and the answer space. A seed dataset $\mathcal{D}_{\text{seed}}$ contains triples $(q, r^\star, a^\star)$, where $q \in \mathcal{Q}$ is a math problem, $r^\star \in \mathcal{R}$ is a reference solution trace, and $a^\star \in \mathcal{A}$ is the reference answer. The student model $S : \mathcal{Q} \to \mathcal{R}$ produces a trace $r_s = S(q)$, and an answer extractor $A : \mathcal{R} \to \mathcal{A}$ recovers the final answer $a_s = A(r_s)$.

An equivalence judge $J : \mathcal{A} \times \mathcal{A} \to \{0, 1\}$ checks whether two answers match under fixed normalization. The student's failure set on $\mathcal{D}_{\text{seed}}$ is $\mathcal{F} = \{(q, r_s, r^\star) : (q, r^\star, a^\star) \in \mathcal{D}_{\text{seed}},\ r_s = S(q),\ J(A(r_s), a^\star) = 0\}$. The reference trace $r^\star$ is included because it is consulted by the teacher during diagnosis.

Specification. We use *specification* to mean the information given to the generator before it writes a new problem. A specification can be enriched in two ways: in granularity, by revealing more about each failure, and in composition, by asking one problem to repair several failures. In DDS, each specification is constructed after diagnosis from a pair of failure cases. It contains the original problems, the student's incorrect solution traces, and the teacher's trace-level diagnoses. Concretely, for two diagnosed failures $(q_i, r_{s,i}, d_i)$ and $(q_j, r_{s,j}, d_j)$, we write $\sigma = \big((q_i, r_{s,i}, d_i),\ (q_j, r_{s,j}, d_j)\big)$. The generator is conditioned on σ to produce a single new problem that targets both diagnoses jointly.

Objective. Given $\mathcal{F}$, our goal is to construct a patch dataset $\mathcal{D}_{\text{patch}} \subset \mathcal{Q} \times \mathcal{R}$ that, when used for supervised fine-tuning of the student, recovers the failures in $\mathcal{F}$ and generalizes to held-out problems. We constrain $\mathcal{D}_{\text{patch}}$ to a fixed budget so that comparisons across synthesis methods are at equal size. Each accepted patch must pass quality control for well-posedness and solution correctness.

3.2 The DDS framework

DDS turns pairs of failures in $\mathcal{F}$ into synthesized patches through three stages. The teacher T, generator G, and verifier V are three roles, instantiated by capable models appropriate to the modality (specific choices reported in Appendix D.1); full prompt details are in Appendix J.

Stage 1: Trace-centric diagnosis. For each $(q, r_s, r^\star) \in \mathcal{F}$, the teacher inspects the student's trace alongside the reference trace and emits a trace-level diagnosis d. The diagnosis identifies where the student's reasoning went wrong and what constraint, inference, or concept a new problem should require. Diagnoses that only restate that the answer is wrong or only flag formatting issues are discarded. $\mathcal{F}_{\text{diag}} = \{(q, r_s, d) : (q, r_s, r^\star) \in \mathcal{F},\ d = T(q, r_s, r^\star) \text{ is actionable}\}$. We condition on the trace rather than on the outcome label or a capability node because the trace localizes a specific reasoning failure, while an outcome label or capability node would collapse many different trace-level errors into one signal.

Stage 2: Patch synthesis with diagnostic crossover. Given $\mathcal{F}_{\text{diag}}$, the generator produces new problems by composing pairs of diagnosed failures drawn uniformly at random. Conditioned on a pair $(q_i, r_{s,i}, d_i)$ and $(q_j, r_{s,j}, d_j) \in \mathcal{F}_{\text{diag}}$, the generator produces a single coherent problem whose solution requires satisfying both repair requirements simultaneously: $(q', r') \sim G\big(\cdot \mid q_i, r_{s,i}, d_i;\ q_j, r_{s,j}, d_j\big)$. We call this composition step *diagnostic crossover*. The generator is instructed to produce one integrated problem, rather than two loosely concatenated subproblems. The motivation is that repairing diagnosed weaknesses separately may not teach the student how to coordinate them in a single reasoning chain. For example, separate patches may exercise the probability-simplification step and the shaded-region geometry of Figure 2 individually, yet still leave the student unable to solve a problem that requires both at once. We test this distinction empirically in Section 4.4, where we compare diagnostic crossover against a single-failure variant and analyze the gap in Theorem 4.1.

Stage 3: Quality control. Every candidate (q', r', a') from Stage 2, where $a' = A(r')$, is checked by the verifier V. The verifier checks two conditions: well-posedness, meaning that the problem is fully specified and admits a unique answer under the stated constraints; and solution correctness, meaning that the verifier's independent solution to q' matches a'. Candidates passing both checks form the patch dataset: $\mathcal{D}_{\text{patch}} = \{(q', r') : V \text{ accepts } (q', r', a')\}$. The filter is necessary because generated problems may be ill-posed or paired with incorrect solutions.

Fine-tuning and iterative patching. The student is fine-tuned on $\mathcal{D}_{\text{patch}}$ with standard SFT under the same hyperparameters as all baselines (Section 4.1). Because fine-tuning shifts the student's failure distribution, the same DDS procedure can be re-applied to the updated student, producing a sequence of patched students and patch datasets. We study this iterative loop in Section 4.5.

4 Experiments

We evaluate DDS along four research questions: whether it repairs the student's observed failures, whether the gains transfer to held-out benchmarks, what diagnostic crossover adds, and whether re-

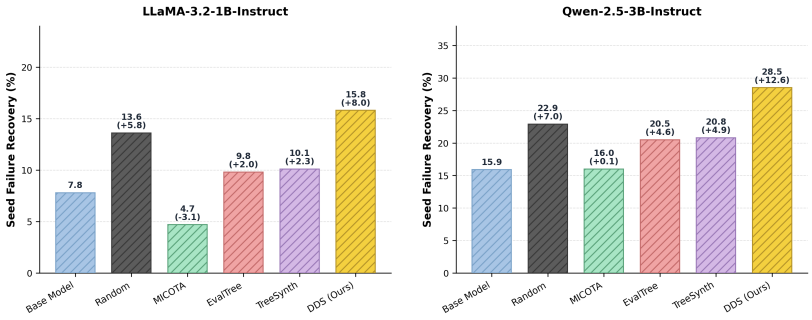


Figure 3: **Seed-failure recovery (Pass@1, %, on the incorrect-seed subset)** on Llama-3.2-1B-Instruct and Qwen2.5-3B-Instruct. Numbers above bars show absolute recovery and the change Δ relative to the base student. DDS achieves the largest gain on both Qwen2.5-3B-Instruct ($+12.6$) and Llama-3.2-1B-Instruct ($+8.0$).

diagnosis remains useful after the student changes. Additional robustness checks and patch-alignment analyses are summarized in the relevant sections and reported in full in the appendix.

(RQ1): Does conditioning synthesis on trace-level diagnoses recover the student's own failures more effectively than sampling or coarser synthesis baselines? (Section 4.2) **(RQ2):** Do these gains transfer to held-out benchmarks beyond the original failure distribution? (Section 4.3) **(RQ3):** What does diagnostic crossover add beyond a single-failure variant, and does patching introduce side effects on problems the student already answered correctly? (Section 4.4) **(RQ4):** As the failure distribution shifts after patching, does fresh re-diagnosis outperform reusing stale patches? (Section 4.5)

4.1 Experimental setup

Seed set and patch budget. The seed set contains 15,010 math problems sampled from AIME, MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and OpenMathReasoning (Moshkov et al., 2025). For each student, we run inference on the seed, collect failures, and synthesize a 12K patch dataset. Throughout the paper, **DDS (ours)** refers to the framework introduced in Section 3, in which every patch is produced by diagnostic crossover. The patch-composition ablation in Section 4.4 introduces a single-failure variant, denoted DDS-Aug, only for that comparison.
Held-out benchmarks. We evaluate on six math benchmarks spanning a wide difficulty range: GSM8K, MATH, AIME 2025, AMC, OlympiadBench (He et al., 2024), and CN-Math-2024. For the three benchmarks that share a source with the seed set, exact-match and 13-gram near-duplicate checks find no overlap with the seed set or the patch set (Appendix D.5).
Baselines. We compare DDS against four baselines: (i) Random (Moshkov et al., 2025), which samples 12K problems from OpenMathReasoning; (ii) EvalTree (Zeng et al., 2025), capability-tree-guided synthesis; (iii) TreeSynth (Wang et al., 2025), coverage-driven synthesis; and (iv) MICOTA (Ding et al., 2025), which adapts teacher responses to the student model.
Reference solutions. For DDS, EvalTree, and TreeSynth, supervision traces are generated by QwQ (Qwen Team, 2025). MICOTA produces student-adapted teacher responses, and Random uses the pre-provided OpenMathReasoning traces, generated by QwQ and DeepSeek (Guo et al., 2025a). Training hyperparameters are held fixed across methods (Appendix C). We also test robustness to teacher choice by replacing QwQ-32B with Qwen3-8B for DDS, EvalTree, and TreeSynth; DDS retains the largest seed-failure recovery and held-out average, with full results in Appendix F.2.
Evaluation. We sample $N = 8$ generations per problem and report Pass@1 in the main text; Pass@8 results are in Appendix E.

4.2 Effectiveness of failure patching (RQ1)

We first ask whether DDS recovers the student's own failures more effectively than the baselines at equal patch budget. The metric is Pass@1 on the subset of seed problems the base student answered incorrectly. Figure 3 summarizes recovery after one-pass fine-tuning on 12K patch examples. **Per-student recovery.** On Qwen2.5-3B-Instruct, DDS achieves the largest gain in seed-failure recovery ($+12.6$; $15.9 \rightarrow 28.5$), exceeding Random ($+7.0$), TreeSynth ($+4.9$), EvalTree ($+4.6$), and MICOTA

Table 1: **Held-out math benchmarks** (Pass@1; $T = 0.8$, $p = 0.95$, $N = 8$). Best per student in **bold**. DDS (ours) achieves the highest average on both students.

Method	GSM8K	MATH	AIME25	AMC	OLYMPIAD	CN-MATH24	Avg
Llama-3.2-1B-Instruct							
Base model	5.2	14.9	0.0	10.6	4.1	1.7	6.08
Random (Moshkov et al., 2025)	31.8	20.2	0.0	8.4	5.2	1.7	11.22
MICOTA (Ding et al., 2025)	8.4	2.8	0.0	4.1	1.1	0.0	2.73
EvalTree (Zeng et al., 2025)	38.8	15.6	0.0	5.6	2.4	0.8	10.53
TreeSynth (Wang et al., 2025)	32.2	17.1	0.0	8.1	3.4	0.8	10.27
DDS (ours)	**43.9**	**22.3**	0.0	**12.2**	**5.9**	**2.1**	**14.40**
Qwen2.5-3B-Instruct							
Base model	83.4	64.1	2.0	37.2	26.2	12.0	37.48
Random (Moshkov et al., 2025)	82.1	65.2	6.3	36.2	29.0	15.4	39.03
MICOTA (Ding et al., 2025)	83.0	63.2	1.7	39.1	27.6	8.3	37.15
EvalTree (Zeng et al., 2025)	84.9	67.9	**11.3**	44.1	32.1	12.9	42.20
TreeSynth (Wang et al., 2025)	86.1	67.4	8.8	43.8	33.3	12.1	41.92
DDS (ours)	**87.6**	**71.9**	10.4	**46.2**	**37.3**	**16.7**	**45.02**

(+0.1). The comparison against EvalTree is the central result: both methods start from the same student failures, but DDS conditions synthesis on their traces while EvalTree reduces them to capability labels, and the recovery gain of DDS is about 2.7× that of EvalTree. On Llama-3.2-1B-Instruct, DDS improves recovery by +8.0 (7.8 → 15.8), while no baseline exceeds +5.8 (Random).

Takeaway 1 (*RQ1: trace-level specifications recover failures more effectively than coarser ones*). At the same patch budget, DDS recovers substantially more seed failures than capability-profiling and coverage-driven synthesis. The value of failure information depends on the granularity at which it is fed to the generator.

4.3 GENERALIZATION TO HELD-OUT BENCHMARKS (RQ2)

We next ask whether the recovery gains from Section 4.2 transfer beyond the seed distribution. Table 1 reports Pass@1 across the six held-out benchmarks. On Qwen2.5-3B-Instruct, DDS improves the math average from 37.48 to 45.02 (+7.54), exceeding every baseline (EvalTree by +2.82, TreeSynth by +3.10), with gains on every benchmark. On Llama-3.2-1B-Instruct, DDS improves the math average from 6.08 to 14.40 (+8.32), with gains on MATH (14.9 → 22.3), AMC (10.6 → 12.2), OlympiadBench (4.1 → 5.9), and CN-Math-2024 (1.7 → 2.1). All methods score 0.0 on AIME25 at this scale, suggesting a capability ceiling under this model scale and patch budget. DDS is also stable across independent runs: over three runs on Qwen2.5-3B-Instruct, seed-failure recovery is 28.48 ± 0.05 and the held-out math average is 44.78 ± 0.21; full per-benchmark variance is in Appendix F.1. Its lead over every baseline persists with only 7K patches or without verification (Appendix D).

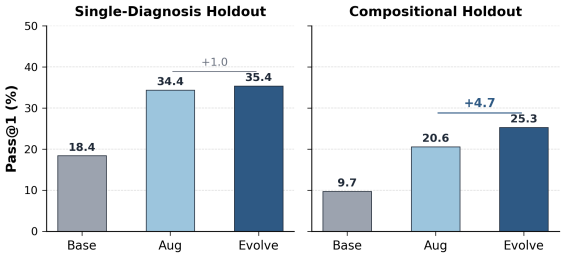
Takeaway 2 (*RQ2: gains transfer beyond the seed*). DDS improves held-out math performance on both students, exceeding every baseline by at least +3.18 (1B) and +2.82 (3B) on the average, indicating that the gain reflects transferable supervision rather than overfitting to the original failures.

4.4 WHAT CROSSOVER ADDS, AND SIDE EFFECTS OF PATCHING (RQ3)

This subsection isolates diagnostic crossover by comparing the default DDS against a single-failure variant, and then checks whether patching introduces side effects on problems the student already answered correctly.

Method	Specification	Seed fail.	Math avg
Base model	–	15.9	37.48
TreeSynth	problem pool	20.8	41.92
EvalTree	capability label	20.5	42.20
DDS-AUG	one trace diagnosis	26.2	44.57
DDS-EVOLVE	two trace diagnoses	**28.5**	**45.02**

(a) Aggregate metrics by specification.



(b) Single-diagnosis and compositional holdouts.

Figure 4: **Diagnostic crossover ablation.** Two complementary views of the same finding. **(a)** A single trace-level diagnosis (DDS-AUG) already exceeds EvalTree and TreeSynth, and composing two diagnoses (DDS-EVOLVE) improves both seed-failure recovery and the held-out math average further. **(b)** The two variants are tied on problems that test one diagnosed skill in isolation, while DDS-EVOLVE pulls ahead on problems that require two diagnosed skills used together in one solution — locating the specific regime where crossover supervision matters.

Single-failure variant. For this ablation only, we instantiate a single-failure variant of DDS, denoted DDS-AUG. It uses a single-failure specification $\sigma_{\text{aug}} = (q, r_s, d)$ – one diagnosed failure – in place of the pair-of-failures specification σ defined in Section 3.1. We refer to the default crossover-inclusive variant as DDS-EVOLVE only in this subsection. Diagnosis (Stage 1) and verification (Stage 3) are unchanged; the two variants differ only in the form of the specification fed to the generator at Stage 2. Without composition, DDS-AUG already exceeds EvalTree and TreeSynth (Figure 4a), so the advantage of DDS does not come from crossover alone.

Crossover helps most when failures interact. We compare DDS-EVOLVE and DDS-AUG on Qwen2.5-3B-Instruct under the same fine-tuning budget, in two complementary views (Figure 4). DDS-EVOLVE improves seed-failure recovery from 26.2 to 28.5 and the held-out math average from 44.57 to 45.02 over DDS-AUG. Both metrics move in the same direction, indicating that the crossover-inclusive specification is at least as useful as the single-failure specification on every aggregate measure we report. To localize where the gain originates, we construct two holdout sets, drawn from the same generators as the training patches but never used for training. The *single-diagnosis holdout* contains problems that each test one diagnosed skill in isolation; the *compositional holdout* contains problems whose solution requires using two diagnosed skills together within one chain of reasoning. On the single-diagnosis holdout, DDS-EVOLVE and DDS-AUG are nearly tied (35.4 vs. 34.4): single-failure patches already give the student adequate practice on isolated skills. On the compositional holdout, DDS-EVOLVE pulls ahead by $+4.7$ (25.3 vs. 20.6). The two views agree on the same finding: crossover supervision is what teaches the student to use two diagnosed skills together in one solution, rather than each one in isolation.

A linear explanation. The compositional holdout result is consistent with a simple linear analysis of teacher–student distillation. Following work that recovers concepts, tasks, and functions from approximately linear directions in LLM representations (Park et al., 2024; Hendel et al., 2023; Todd et al., 2024; Wu et al., 2024) and work on latent composition subspaces (Lee et al., 2025; Song et al., 2025), we distinguish atomic directions from interaction directions. Single-failure patches expose individual diagnoses, while crossover patches additionally expose directions that appear only when two diagnoses must be coordinated.

Theorem 4.1 (Informal: compositional patches reduce a residual risk)**.** *Let $\mathcal{D}_{\text{simp}}$ contain only single-failure examples, and let $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{simp}} \cup \mathcal{D}_{\text{comp}}$ additionally contain compositional examples that activate interaction subspaces $\mathcal{V}_S$. Then single-failure training leaves a compositional risk floor that compositional examples can reduce:*

$$R(\mathcal{D}_{\text{simp}}) \;\geq\; \sum_{\substack{S \subseteq [M] \\ |S| \geq 2}} \left\| \Delta^{(S)} \right\|_2^2, \qquad 0 \;\leq\; R(\mathcal{D}_{\text{mix}}) \;\leq\; R(\mathcal{D}_{\text{simp}}). \tag{1}$$

The full setup, assumptions, and proof are in Appendix B. In our setting, DDS-AUG corresponds to single-failure supervision, $\mathcal{D}_{\text{simp}}$, while DDS-EVOLVE, corresponding to $\mathcal{D}_{\text{mix}}$, additionally

Table 2: **Iterative-patching results (Pass@1) on Qwen2.5-3B-Instruct.** DDS_1 reuses the iteration-1 patch dataset; DDS_2 re-diagnoses the updated student and synthesizes a fresh patch dataset for iteration 2. Random samples a fresh 12K subset for iteration 2 as a non-adaptive baseline.

Method	SEED FAILURE	GSM8K	MATH	AIME25	AMC	OLYMPIAD	CN-MATH24	Avg
Random (Iter2)	28.6	85.1	72.1	12.5	48.8	35.0	15.4	44.8
DDS_1 (reuse Iter1)	30.7	87.1	72.7	14.7	**50.9**	37.1	12.5	45.8
DDS_2 (fresh Iter2)	**33.1**	**87.9**	**74.6**	**15.4**	49.3	**39.8**	**21.3**	**48.1**

exposes pairs of interacting diagnoses. The $+4.7$ compositional-holdout gap is consistent with this residual-risk interpretation.

Side effects and patch alignment. Patching has limited side effects on problems the student initially answered correctly: on a 500-problem diagnostic subset, only 9 cases (1.8%) drop from Pass@8 ≥ 1 before fine-tuning to $0/8$ after fine-tuning; the full distribution is in Appendix F.4. We also run a patch–improvement alignment analysis: for problems improved by a fine-tuned model, we retrieve the most similar examples from that model's 12K patch set. DDS produces both the most improved problems and the highest top-500 patch similarity across MATH, OlympiadBench, and seed failures; full setup and results are in Appendix G.

Takeaway 3 (*RQ3: crossover targets failures that interact*). On aggregate, DDS-EVOLVE improves seed-failure recovery and the held-out math average over the single-failure variant. This pattern is clearer on the controlled holdouts: when a problem requires only one diagnosed skill, the two variants perform similarly; when a problem requires two diagnosed skills to be used together, DDS-EVOLVE leads by $+4.7$ Pass@1. Our linear residual-risk analysis matches this pattern, and side effects on initially-correct problems are limited (1.8% new failures).

4.5 ITERATIVE PATCHING (RQ4)

After one round of patching, the student's failure distribution shifts: some previously failed problems become solvable, while new failure patterns can emerge. A practitioner has two natural follow-up strategies, and we compare both starting from the iteration-1 patched student.

Setup. DDS_1 continues training on the iteration-1 patch dataset (data reuse). DDS_2 re-runs trace-centric diagnosis on the updated student, synthesizes a fresh patch dataset including diagnostic crossover, and fine-tunes on the new patches. As a non-adaptive baseline, we re-train Random on a freshly sampled 12K subset of OpenMathReasoning for iteration 2.

Results. Re-diagnosis yields consistent additional gains over data reuse (Table 2). Seed-failure recovery rises from 28.6 (Random Iter2) to 30.7 under DDS_1 and to 33.1 under DDS_2; the math average rises from 44.8 to 45.8 to 48.1. The largest incremental gains under DDS_2 appear on the harder benchmarks: MATH (72.7 $\rightarrow$ 74.6), AIME25 (14.7 $\rightarrow$ 15.4), OlympiadBench (37.1 $\rightarrow$ 39.8), and CN-Math-2024 (12.5 $\rightarrow$ 21.3). Because the patched student has a different failure distribution from the iteration-0 student, the iteration-1 patches are no longer aligned with what the model still needs to repair; fresh re-diagnosis re-aligns the patches with the new failure distribution. This suggests that iterative patching should refresh the diagnostic signal after each student update, rather than reuse patches generated for an earlier failure distribution.

Takeaway 4 (*RQ4: re-diagnosis beats reusing stale patches*). As the failure distribution shifts after patching, re-running diagnosis and synthesizing fresh patches yields larger second-round gains than continuing to train on the original patch set, supporting an iterative deployment loop.

4.6 CASE STUDY: EXTENSION TO VISION-LANGUAGE DATA

DDS is modality-agnostic at the framework level: any setting in which a student attempt can be diagnosed and a candidate patch can be verified admits the same loop. To examine this, we extend DDS to vision-language post-training. We use Qwen2.5-VL-7B-Instruct as the student, GPT-5.2 as the teacher for diagnosis and reference generation, and Qwen3.5-27B-Instruct for patch synthesis. Following the setup in Section 4.1, we replace the text-only seed set with the multimodal dataset of

Chen et al. (2025). We evaluate on a diverse suite of held-out benchmarks via VLMEvalKit (Duan et al., 2024): OCRBench, MathVista$_{\text{MINI}}$, MathVision$_{\text{MINI}}$, MMVet, HallusionBench, DynaMath, and RealWorldQA, covering OCR, mathematical reasoning, visual understanding, hallucination robustness, dynamic reasoning, and real-world image understanding. DDS achieves the highest

Table 3: **Multimodal benchmark results** on seven held-out benchmarks. Best in **bold**.

Method	OCR	MATHVISTA	MATHVISION	MMVET	HALLUSION	DYNAMATH	REALWORLD	Avg
Base model	77.6	54.2	22.5	63.0	38.9	40.4	52.6	49.89
Random	80.2	56.8	23.4	66.8	39.4	41.9	53.7	51.74
TreeSynth	83.2	67.1	27.5	71.1	41.5	48.8	64.5	57.67
DDS (ours)	**84.2**	**69.5**	**29.3**	**71.3**	**47.2**	**52.9**	**65.1**	**59.93**

average (59.93) among the methods we compare (Table 3; confidence intervals in Appendix F.3). Many vision-language errors are also traceable to specific failure modes: a model may misread text in an image, attend to the wrong region, miss a spatial relation, ignore a visual constraint, or combine correct perception with incorrect downstream reasoning. DDS turns these failures into targeted multimodal patches, and diagnostic crossover further produces examples that require multiple visual and reasoning skills to be repaired jointly. Compared with TreeSynth, which broadens question diversity, DDS stays anchored to the student's actual failure traces and produces patches that are more directly aligned with its weaknesses. Qualitative examples are in Appendix I: TreeSynth tends to generate relatively basic questions close to the seed pool, whereas DDS produces more challenging crossover-style questions that combine multiple diagnosed weaknesses in a single multimodal instance.

Takeaway 5 (*Extension to other modalities*). DDS *extends* to vision-language post-training, exceeding the strongest baseline, TreeSynth, by $+2.26$ on a seven-benchmark multimodal average.

5 CONCLUSION

We introduced DDS, a framework that turns a student model's own failures into targeted SFT supervision. DDS first performs *Trace-Centric Diagnosis*, which extracts an actionable repair directive from the student's incorrect solution trace, and then applies *Diagnostic Crossover*, which composes two such diagnoses into a single problem requiring joint repair. Across two student models and six math reasoning benchmarks, DDS outperforms capability profiling and coverage-driven synthesis at the same patch budget. The gains also transfer to held-out benchmarks, showing that DDS does more than narrowly recover the original seed failures. Iterative re-diagnosis provides further gains as the student's failure distribution shifts, suggesting that diagnostic signals should be refreshed after each round of patching. The same loop extends to vision-language post-training, where DDS reaches the highest average across seven multimodal benchmarks, suggesting that the framework's value depends on having a diagnosable failure and a verifiable patch rather than on the modality. Overall, DDS provides a model-aware mechanism for improving reasoning models with verifiable training data synthesized directly from their own failure traces.

6 LIMITATIONS

Budgeted iterative patching. DDS naturally supports multiple patching rounds, but we evaluate only two iterations because each round requires re-diagnosis, re-synthesis, and verification. We do not optimize how to allocate the training budget across rounds, such as epochs per round or the total number of iterations; this remains future work.

Depth of diagnostic crossover. Our crossover operator composes two diagnoses into one patch. Deeper crossover could target richer interactions, but may also increase generation cost and the risk of ill-posed problems. Studying this trade-off is a promising direction.

Computational cost. DDS adds diagnosis, synthesis, and verification on top of standard SFT, requiring additional teacher or verifier calls (per-stage counts in Appendix D.2). This overhead may limit use in budget-constrained settings; distilling these roles into smaller specialized models is a natural way to reduce cost.

AI USE STATEMENT

Besides the LLMs that generate synthetic training data as part of the proposed method (Section 3 and Appendix D.1), we used generative AI tools for English grammar and editing of the paper, and the authors take full responsibility for its content.

ETHICS STATEMENT

This work uses existing benchmark datasets, openly available or cited models, synthetic data generation, and automated evaluation; it does not involve human subjects or the collection of sensitive personal data. DDS is a general method for targeted reasoning-data synthesis rather than a deployed system, so its societal impacts are indirect. It may make model improvement more targeted and data-efficient, but, as with any capability-improving synthesis method, it could be misused in unsafe domains, and incorrect diagnosis or insufficient verification could produce flawed training data. These risks can be mitigated by restricting high-risk domains, using independent verification and human review where appropriate, and evaluating downstream behavior before deployment.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Shengnan An, Zexiong Ma, Siqi Cai, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Can LLMs learn from mistakes? an empirical study on reasoning tasks. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 833–854, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.46. URL https://aclanthology.org/2024.findings-emnlp.46/.

Yan Chen, Long Li, Teng Xi, Long Zeng, and Jingdong Wang. Perception before reasoning: Two-stage reinforcement learning for visual reasoning in vision-language models. *arXiv preprint arXiv:2509.13031*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cuong Dang, Dung D. Le, and Thai Le. A curious case of searching for the correlation between training data and adversarial robustness of transformer textual models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13475–13491, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.800. URL https://aclanthology.org/2024.findings-acl.800/.

Dongyi Ding, Tiannan Wang, Chenghao Zhu, Meiling Tao, Yuchen Eleanor Jiang, and Wangchunshu Zhou. MiCoTA: Bridging the learnability gap with intermediate CoT and teacher assistants. *arXiv preprint arXiv:2507.01887*, 2025.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pp. 11198–11201, 2024.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20640–20661. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/guan25f.html.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

Honglin Guo, Kai Lv, Qipeng Guo, Tianyi Liang, Zhiheng Xi, Demin Song, Qiuyinzhe Zhang, Yu Sun, Kai Chen, Xipeng Qiu, and Tao Gui. CritiQ: Mining data quality criteria from human preferences. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16240–16261, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.792. URL https://aclanthology.org/2025.acl-long.792/.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Simran Kaur, Simon Park, Anirudh Goyal, and Sanjeev Arora. Instruct-SkillMix: A powerful pipeline for LLM instruction tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=44z7HL4mfX.

Jin Hwa Lee, Thomas Jiralerspong, Lei Yu, Yoshua Bengio, and Emily Cheng. Geometric signatures of compositionality across a language model's lifetime. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5292–5320, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.265. URL https://aclanthology.org/2025.acl-long.265/.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*, 2023. URL https://mlanthology.org/tmlr/2023/liang2023tmlr-holistic/.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7Jwpw4qKkb.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Vishvak Murahari, Ameet Deshpande, Peter Clark, Tanmay Rajpurohit, Ashish Sabharwal, Karthik Narasimhan, and Ashwin Kalyan. QualEval: Qualitative evaluation for model improvement. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2093–2111, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.115. URL https://aclanthology.org/2024.naacl-long.115/.

Vinh Nguyen, Cuong Dang, Jiahao Zhang, Hoa Tran, Minh Tran, Trinh Chau, Thai Le, Lu Cheng, and Suhang Wang. URAG: A benchmark for uncertainty quantification in retrieval-augmented large language models. *arXiv preprint arXiv:2603.19281*, 2026.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H. Vicky Zhao, Conghui He, and Lijun Wu. LEMMA: Learning from errors for MatheMatical advancement in LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11615–11639, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.605. URL https://aclanthology.org/2025.findings-acl.605/.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/, March 2025.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Nicolaus Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024. URL https://openreview.net/forum?id=p09i9IW9eJ.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR Tulu: Reinforcement learning with evolving rubrics for deep research. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=97NEP1pyS3.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Jiajun Song, Zhuoyan Xu, and Yiqiao Zhong. Out-of-distribution generalization via composition: a lens through induction heads in transformers. *Proceedings of the National Academy of Sciences*, 122(6):e2417182122, 2025.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Minh Tran, Cuong Dang, Tuc Nguyen, Khanh-Tung Tran, Minh Huynh Nguyen, Trinh Chau, Kien Le, Do Xuan Long, Jiahao Zhang, Fali Wang, et al. Retrieved but not reliable: A survey on attacks, and defenses in retrieval-augmented generation. *arXiv preprint arXiv:2608.24977*, 2026.

Sheng Wang, Pengan Chen, Jingqi Zhou, Qintong Li, Jingwei Dong, Jiahui Gao, Boyang Xue, Jiyue Jiang, Lingpeng Kong, and Chuan Wu. TreeSynth: Synthesizing diverse data from scratch via tree-guided subspace partitioning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 63870–63918. Curran Associates, Inc., 2025. doi: 10.52202/

085713-2144. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5c641b1c7e647c8ad9f62431960b3e1a-Paper-Conference.pdf.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=fykjplMc0V.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=CfXh93NDgH.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=yLClGs770I.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488. Curran Associates, Inc., 2022. doi: 10.52202/068431-1126. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/639a9a172c044fbb64175b5fad42e9a5-Paper-Conference.pdf.

Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. EvalTree: Profiling language model weaknesses via hierarchical capability trees. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=aV2hQN9vkp.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

APPENDIX OVERVIEW

Table 4: **Appendix roadmap.** Each appendix section expands a corresponding part of the main paper.

Appendix	What it contains
Appendix A	Extended related work on synthesis specifications, weakness profiling, mistake correction, and reflective optimization.
Appendix B	Full setup, assumptions, and proof for the compositional-risk interpretation used in Section 4.4.
Appendix C	Fine-tuning configuration, including hardware, learning rate, epochs, and batch sizes.
Appendix D	Model roles, construction cost, a reduced-budget comparison, verification ablation, and decontamination checks.
Appendix E	Additional math benchmark tables, including Pass@8 results and longer-schedule robustness checks.
Appendix F	Robustness and side-effect analyses: run-to-run variance, teacher-model robustness, multimodal evaluation variance, and initially-correct problem degradation.
Appendix G	Full patch–improvement alignment analysis, including retrieval setup, improved-problem counts, and similarity results.
Appendix H	Full worked crossover example with parent problems, diagnoses, targeted syntheses, and the final crossover problem.
Appendix I	Qualitative multimodal examples comparing DDS with TREESYNTH.
Appendix J	Diagnosis, diagnostic crossover, and single-failure augmentation prompts.

A Extended Related Work

This appendix expands the discussion in Section 2, covering broader work on instruction synthesis, math reasoning data generation, weakness profiling, mistake correction, reflective optimization, and benchmark-based model analysis.

Specifications in synthetic data generation. Synthesis methods for LLM post-training differ in what they condition the generator on. We describe these conditioning signals as specifications, and we group prior work by the type of specification it uses. *Student-agnostic specifications* pass only task descriptions, seed pools, or high-level skills extracted from such sources to the generator, without reference to any particular student. WizardLM (Xu et al., 2024) rewrites prompts to deepen reasoning, MetaMath (Yu et al., 2024) rewrites mathematical questions from multiple perspectives, MAmmoTH (Yue et al., 2024) trains on a curated mixture of math instruction datasets with chain-of-thought and program-of-thought rationales, rStar-Math (Guan et al., 2025) bootstraps reasoning traces through self-evolution, and TreeSynth (Wang et al., 2025) recursively partitions the data space to broaden coverage. STaR (Zelikman et al., 2022) and ReST (Gulcehre et al., 2023) similarly bootstrap rationales by filtering on outcome correctness. Parallel work in safety (Liu et al., 2024; Samvelyan et al., 2024) adopts coverage-driven evolutionary search to discover diverse adversarial prompts. Closer to our compositional generation step, Instruct-SkillMix (Kaur et al., 2025) extracts a pool of natural-language skills from a powerful LLM and conditions generation on a randomly chosen pair to promote diversity; the skill pool, however, is built from task and instruction sources, not from any particular student's failures, so the random skill pair is also student-agnostic. None of these methods decide what to generate from the reasoning error made by a particular student. *Capability-level specifications* add a coarse, aggregate failure signal: EvalTree (Zeng et al., 2025) attaches failures to nodes in a hierarchical capability taxonomy and synthesizes problems that test the flagged node, while QualEval (Murahari et al., 2024) uses LLM-generated qualitative insights to highlight where a model underperforms. The supervision signal is a label shared across many problems: two failures attached to the same node may have different first invalid inferences, yet the generator receives the same node-level specification. DDS goes one step further: the specification is the student's own failed solution trace, distilled by the teacher into an actionable directive that identifies where the reasoning diverged and what a new problem should require. The pair-of-diagnoses input used by diagnostic crossover may superficially resemble Instruct-SkillMix's skill-pair recipe, but the pair is drawn from the specific student's own diagnosed failures rather than from a fixed, student-agnostic skill pool, so the resulting problem targets that student's reasoning errors rather than widening coverage in general.

Failure-driven training without synthesis. A complementary line uses failures directly rather than as a generator condition. Mistake-correction methods such as LeMa (An et al., 2024) and LEMMA (Pan et al., 2025) pair a wrong trace with a teacher-corrected one and fine-tune the student to map the original question to the corrected trace. Self-correction work (Madaan et al., 2023; Shinn et al., 2023) elicits revisions at inference time rather than baking them into the weights. These methods establish that errors carry useful information, but their supervision remains anchored to existing or surface-augmented questions, rather than using the diagnosed failure as a condition for synthesizing a new problem that composes repair requirements. DDS differs in that the failure trace conditions the synthesis of *new* problems, including, through diagnostic crossover, problems that compose two failures into one, rather than relabeling the original ones.

Reflective optimization with verbalized feedback. A neighboring line optimizes prompts or rubrics iteratively using natural-language feedback rather than weight updates. OPRO (Yang et al., 2024) treats the LM as an optimizer over instruction strings; TextGrad (Yuksekgonul et al., 2025) backpropagates textual feedback through compound systems; CritiQ (Guo et al., 2025b) mines verbalized data-quality criteria from human preferences; GEPA (Agrawal et al., 2026) extends reflective prompt evolution with Pareto-based candidate selection; DR Tulu (Shao et al., 2026) maintains rubrics that co-evolve with the policy during reinforcement-learning training. These methods do close a verbalized feedback loop, but the natural-language signal conditions a prompt, an instruction, or a reward rather than the synthesis of new training problems. At a high level, GEPA's observation that the natural-language signal should be refreshed as the system changes also motivates our iterative re-diagnosis procedure (Section 4.5), which re-runs trace diagnosis on the updated student rather than reusing stale patches.

Table 5: Notation for the embedding-space formulation of atomic and compositional distillation.

Notation	Meaning
$\mathcal{X} \subseteq \mathbb{R}^d$	Input embedding space.
d	Total embedding dimension.
M	Number of atomic task groups.
$[M]$	Index set $\{1, \ldots, M\}$.
G_m	The m-th atomic task group.
U_m	Atomic subspace associated with task group G_m.
V_S	Interaction subspace associated with a composition over groups $S \subseteq [M]$.
d_m	Dimension of atomic subspace U_m.
d_S	Dimension of interaction subspace V_S.
P_m	Orthogonal projector onto U_m.
P_S	Orthogonal projector onto V_S.
$x \in \mathcal{X}$	Input embedding.
$x_m = P_m x$	Atomic component of x in subspace U_m.
$x_S = P_S x$	Interaction component of x in subspace V_S.
$A(x)$	Set of active atomic groups for input x.
$w^\star$	Teacher parameter vector.
w_0	Initial student parameter vector.
$\Delta = w^\star - w_0$	Teacher–student knowledge gap.
$\Delta^{(m)}$	Atomic component of Δ in subspace U_m.
$\Delta^{(S)}$	Interaction component of Δ in subspace V_S.
$\mathcal{D}_{\text{simp}}$	Training dataset containing only atomic/simple-task examples.
$\mathcal{D}_{\text{comp}}$	Training dataset containing compositional examples.
$\mathcal{D}_{\text{mix}}$	Mixed dataset $\mathcal{D}_{\text{simp}} \cup \mathcal{D}_{\text{comp}}$.
P_{simp}	Orthogonal projector onto the span of $\mathcal{D}_{\text{simp}}$.
P_{mix}	Orthogonal projector onto the span of $\mathcal{D}_{\text{mix}}$.
$\widehat{w}_{\text{simp}}$	Student parameter after training on $\mathcal{D}_{\text{simp}}$.
$\widehat{w}_{\text{mix}}$	Student parameter after training on $\mathcal{D}_{\text{mix}}$.
$R(\mathcal{D})$	Residual learning risk after training on dataset $\mathcal{D}$.
$R_{\text{comp}}(\mathcal{D})$	Residual risk restricted to compositional interaction subspaces.

Weakness profiling and benchmarks. Holistic profiling tools such as HELM (Liang et al., 2023) establish broad evaluation suites that surface where models fail. Inspired by (Nguyen et al., 2026; Dang et al., 2024; Tran et al., 2026), which shows that LLM failures can be amplified by incorrect or misleading evidence, DDS examines model failures at a finer granularity: rather than treating failure as a final-answer outcome, it analyzes the student's solution trace to identify where the reasoning process breaks and what kind of new problem should be synthesized to repair that weakness.

B Theoretical Analysis

This appendix provides the formal analysis deferred from the informal statement in Theorem 4.1. The main text states that training only on single-failure examples can leave a residual compositional risk floor, while adding crossover-style compositional examples can reduce this floor. We make this claim precise through an embedding-space distillation model in which the teacher–student gap decomposes into atomic task components and higher-order interaction components.

The analysis proceeds in three steps. First, Theorem B.5 shows that a dataset containing only atomic examples, $\mathcal{D}_{\text{simp}}$, cannot close the gap along interaction subspaces and therefore leaves a compositional risk floor. Second, Theorem B.6 shows that adding compositional examples to form $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{simp}} \cup \mathcal{D}_{\text{comp}}$ cannot increase the residual learning risk, and strictly reduces it whenever the new examples span previously uncovered components of the teacher–student gap. Finally, Corollary B.7 states the limiting case: when the mixed dataset spans the relevant interaction components, the compositional risk floor of $\mathcal{D}_{\text{mix}}$ becomes zero. Together, these results formalize the intuition behind diagnostic crossover: compositional patches help because they expose interaction directions that single-failure supervision does not cover.

Definition B.1 (Atomic and compositional task subspaces)**.** Let the input embedding space be $\mathcal{X} \subseteq \mathbb{R}^d$. Assume that the task-relevant embedding space decomposes as

$$\mathbb{R}^d = \left(\bigoplus_{m=1}^{M} \mathcal{U}_m \right) \oplus \left(\bigoplus_{\substack{S \subseteq [M] \\ |S| \geq 2}} \mathcal{V}_S \right).$$

Here, $\mathcal{U}_m$ is the atomic subspace associated with task m, and $\mathcal{V}_S$ is the compositional interaction subspace associated with joint tasks over atomic groups indexed by S.

Thus, every input embedding $x \in \mathcal{X}$ admits a unique decomposition

$$x = \left(\bigoplus_{m=1}^{M} x_m \right) \oplus \left(\bigoplus_{\substack{S \subseteq [M] \\ |S| \geq 2}} x_S \right),$$

where

$$x_m \in \mathcal{U}_m, \qquad x_S \in \mathcal{V}_S.$$

If x belongs to an atomic task in group m, then its task-relevant component is $x_m \in \mathcal{U}_m$. If x belongs to a compositional task over $S \subseteq [M]$, $|S| \geq 2$, then its task-relevant components are

$$\{x_m : m \in S\} \quad \text{and} \quad x_S \in \mathcal{V}_S.$$

That is, a compositional task reuses the atomic subspaces of its component groups and additionally depends on the interaction subspace $\mathcal{V}_S$.

Theorem B.2 (Distillation solution)**.** *Let*

$$\Delta := \mathbf{w}_* - \mathbf{w}_0$$

denote the teacher–student knowledge gap. Let $P_{\mathbf{X}}$ be the orthogonal projector onto the data span $\operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$*. Then, under gradient flow training, as $\tau \to \infty$, the student parameter $\mathbf{w}(\tau)$ converges to*

$$\hat{\mathbf{w}} = \mathbf{w}_0 + P_{\mathbf{X}} \Delta.$$

Equivalently,

$$\hat{\mathbf{w}} = \mathbf{w}_0 + P_{\mathbf{X}} (\mathbf{w}_* - \mathbf{w}_0).$$

Theorem B.3 (Decomposed distillation solution)**.** *Let $\Delta := \mathbf{w}_* - \mathbf{w}_0$ denote the teacher–student knowledge gap. Decompose*

$$\Delta = \left(\bigoplus_{m=1}^{M} \Delta^{(m)} \right) \oplus \left(\bigoplus_{\substack{S \subseteq [M] \\ |S| \geq 2}} \Delta^{(S)} \right), \text{ where } \quad \Delta^{(m)} \in \mathcal{U}_m, \quad \Delta^{(S)} \in \mathcal{V}_S. \tag{2}$$

Let $P_{\mathbf{X}}$ be the orthogonal projector onto $\operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$*. Then, under gradient flow training, as $\tau \to \infty$, the student parameter $\mathbf{w}(\tau)$ converges to*

$$\hat{\mathbf{w}} = \mathbf{w}_0 + P_{\mathbf{X}} \left(\bigoplus_{m=1}^{M} \Delta^{(m)} \right) + P_{\mathbf{X}} \left(\bigoplus_{\substack{S \subseteq [M] \\ |S| \geq 2}} \Delta^{(S)} \right). \tag{3}$$

Corollary B.4 (No benefit from data in already-matched directions)**.** *Let $\mathcal{U} \subseteq \mathbb{R}^d$ be a task subspace, and suppose the student already matches the teacher on $\mathcal{U}$, $P_{\mathcal{U}} \Delta = 0$, where $P_{\mathcal{U}}$ is the orthogonal projector onto $\mathcal{U}$. Let $\mathcal{S}$ be the current training dataset, and let $\mathcal{S}' = \mathcal{S} + \mathcal{U}$ be the enlarged training dataset obtained by adding data whose directions lie only in $\mathcal{U}$, and assume that $\mathcal{U}$ is orthogonal to the span of $\mathcal{S}$. Then the residual learning risk does not decrease:*

$$R(\mathcal{S}') = R(\mathcal{S}). \tag{4}$$

Theorem B.5 (Simple-task distillation leaves a compositional risk floor)**.** *Let $\mathcal{D}_{\text{simp}}$ be a training dataset containing only atomic task examples. Then its learning risk satisfies the lower bound*

$$R(\mathcal{D}_{\text{simp}}) \geq \sum_{\substack{S\subseteq[M]\\|S|\geq 2}} \|\Delta^{(S)}\|_2^2. \tag{5}$$

Proof. By the distillation solution,

$$\hat{\mathbf{w}}_{\text{simp}} = \mathbf{w}_0 + P_{\text{simp}}\Delta.$$

Therefore,

$$\mathbf{w}_* - \hat{\mathbf{w}}_{\text{simp}} = \Delta - P_{\text{simp}}\Delta = (I - P_{\text{simp}})\Delta.$$

Hence,

$$R(\hat{\mathbf{w}}_{\text{simp}}) = \|(I - P_{\text{simp}})\Delta\|_2^2.$$

Since the training data contain only simple-task examples,

$$\text{span}(\mathbf{X}_{\text{simp}}) \subseteq \bigoplus_{m=1}^{M} \mathcal{U}_m.$$

Thus, P_{simp} has no projection onto any compositional interaction subspace $\mathcal{V}_S$. Therefore,

$$P_{\text{simp}}\Delta^{(S)} = 0 \qquad \forall S \subseteq [M],\ |S| \geq 2.$$

Using the orthogonal decomposition of Δ, we have

$$(I - P_{\text{simp}})\Delta = (I - P_{\text{simp}}) \left(\bigoplus_{m=1}^{M} \Delta^{(m)} \right) \oplus \left(\bigoplus_{\substack{S\subseteq[M]\\|S|\geq 2}} \Delta^{(S)} \right).$$

By orthogonality,

$$R(\hat{\mathbf{w}}_{\text{simp}}) = \left\| (I - P_{\text{simp}}) \left(\bigoplus_{m=1}^{M} \Delta^{(m)} \right) \right\|_2^2 + \sum_{\substack{S\subseteq[M]\\|S|\geq 2}} \|\Delta^{(S)}\|_2^2.$$

Since the first term is nonnegative,

$$R(\hat{\mathbf{w}}_{\text{simp}}) \geq \sum_{\substack{S\subseteq[M]\\|S|\geq 2}} \|\Delta^{(S)}\|_2^2.$$

□

Theorem B.6 (Compositional data can reduce residual learning risk)**.** *Let $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{simp}} \cup \mathcal{D}_{\text{comp}}$ be a training dataset containing both atomic and compositional examples. Then training on $\mathcal{D}_{\text{mix}}$ leaves no larger residual risk than training on $\mathcal{D}_{\text{simp}}$:*

$$R(\mathcal{D}_{\text{mix}}) \leq R(\mathcal{D}_{\text{simp}}). \tag{6}$$

Moreover, if $\mathcal{D}_{\text{comp}}$ spans a nonzero component of the teacher–student gap that is not captured by $\mathcal{D}_{\text{simp}}$, then the inequality is strict:

$$R(\mathcal{D}_{\text{mix}}) < R(\mathcal{D}_{\text{simp}}). \tag{7}$$

Proof. By the distillation solution,

$$\hat{\mathbf{w}}_{\text{simp}} = \mathbf{w}_0 + P_{\text{simp}}\Delta, \qquad \hat{\mathbf{w}}_{\text{mix}} = \mathbf{w}_0 + P_{\text{mix}}\Delta.$$

Therefore,

$$R(\hat{\mathbf{w}}_{\text{simp}}) = \|\mathbf{w}_* - \hat{\mathbf{w}}_{\text{simp}}\|_2^2 = \|(I - P_{\text{simp}})\Delta\|_2^2,$$

and

$$R(\hat{\mathbf{w}}_{\text{mix}}) = \|\mathbf{w}_* - \hat{\mathbf{w}}_{\text{mix}}\|_2^2 = \|(I - P_{\text{mix}})\Delta\|_2^2.$$

Since

$$\mathcal{S}_{\text{simp}} \subseteq \mathcal{S}_{\text{mix}},$$

projection onto $\mathcal{S}_{\text{mix}}$ leaves no larger residual than projection onto $\mathcal{S}_{\text{simp}}$. Hence,

$$\|(I - P_{\text{mix}})\Delta\|_2^2 \leq \|(I - P_{\text{simp}})\Delta\|_2^2.$$

Thus,

$$R(\hat{\mathbf{w}}_{\text{mix}}) \leq R(\hat{\mathbf{w}}_{\text{simp}}).$$

If there exists

$$u \in \mathcal{S}_{\text{mix}} \cap \mathcal{S}_{\text{simp}}^{\perp}$$

with

$$\langle u, \Delta \rangle \neq 0,$$

then $\mathcal{S}_{\text{mix}}$ captures a nonzero component of the residual gap left by $\mathcal{S}_{\text{simp}}$. Therefore,

$$\|(I - P_{\text{mix}})\Delta\|_2^2 < \|(I - P_{\text{simp}})\Delta\|_2^2,$$

which gives

$$R(\hat{\mathbf{w}}_{\text{mix}}) < R(\hat{\mathbf{w}}_{\text{simp}}).$$

□

Corollary B.7 (Mixed-task distillation can remove the compositional risk floor)**.** *Let $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{simp}} \cup \mathcal{D}_{\text{comp}}$ be the mixed training dataset in Theorem B.6. If the compositional examples span all interaction components of the teacher–student gap, i.e.,*

$$P_{\text{mix}}\Delta^{(S)} = \Delta^{(S)} \qquad \forall S \subseteq [M],\ |S| \geq 2, \tag{8}$$

then the compositional risk floor of $\mathcal{D}_{\text{mix}}$ is zero:

$$R_{\text{comp}}(\mathcal{D}_{\text{mix}}) = \sum_{\substack{S \subseteq [M] \\ |S| \geq 2}} \|(I - P_{\text{mix}})\Delta^{(S)}\|_2^2 = 0. \tag{9}$$

Proof. By Theorem B.5, a dataset containing only atomic examples leaves a residual compositional risk floor because atomic examples do not project onto the interaction subspaces $\mathcal{V}_S$. In contrast, $\mathcal{D}_{\text{mix}}$ includes compositional examples. By assumption, these examples span the interaction components of the teacher–student gap, so

$$P_{\text{mix}}\Delta^{(S)} = \Delta^{(S)} \qquad \forall S \subseteq [M],\ |S| \geq 2. \tag{10}$$

Therefore,

$$(I - P_{\text{mix}})\Delta^{(S)} = 0 \qquad \forall S \subseteq [M],\ |S| \geq 2. \tag{11}$$

Substituting into the definition of the compositional residual risk gives

$$R_{\text{comp}}(\mathcal{D}_{\text{mix}}) = \sum_{\substack{S \subseteq [M] \\ |S| \geq 2}} \|(I - P_{\text{mix}})\Delta^{(S)}\|_2^2 = 0. \tag{12}$$

Thus, mixed-task distillation eliminates the compositional risk floor when the compositional data close the teacher-student gap in the interaction subspaces. □

C TRAINING DETAILS

We fine-tune all models with full-parameter supervised fine-tuning (full SFT) on NVIDIA H100 GPUs. Unless otherwise noted, we use a learning rate of 1×10^{-5}.

For LLAMA (the smaller model), we train for 10 epochs to allow *more learning signal / longer optimization* under the same data budget.

For QWEN-2.5-3B-INSTRUCT, we report results for 5-epoch training runs.

In terms of hardware and batch configuration, QWEN uses 8 H100 GPUs with per-device batch size 1 and gradient accumulation 2, yielding a global batch size of $1 \times 2 \times 8 = 16$. LLAMA uses 4 H100 GPUs with the same per-device batch size 1 and gradient accumulation 2, yielding a global batch size of $1 \times 2 \times 4 = 8$. All other training hyperparameters are kept identical across these settings.

D CONSTRUCTION COST, VERIFICATION, AND DECONTAMINATION

This section reports the model used in each role, the construction cost of DDS and the baselines, a comparison under a reduced patch budget, the effect of verification, and the overlap checks between training and evaluation data. Unless noted otherwise, all results are for Qwen2.5-3B-Instruct.

D.1 MODEL ROLES AND PAIR SELECTION

Table 6 lists the model used in each role. In the math experiments, every method that has a diagnosis, problem-generation, or verification stage runs it with GPT-5, and DDS, EvalTree, and TreeSynth use QwQ-32B for reference solutions, so DDS has access to no stronger model than EvalTree and TreeSynth. At evaluation, the answer extractor and the equivalence judge are rule-based: the answer is parsed from the boxed expression and compared by symbolic equivalence. Diagnostic crossover draws each pair uniformly at random from $\mathcal{F}_{\text{diag}}$, without topic, category, or compatibility constraints.

Table 6: Models used in each role.

Role	**Math**	**Multimodal**
Diagnosis	GPT-5	GPT-5.2
Problem generation	GPT-5	Qwen3.5-27B-Instruct
Reference solutions	QwQ-32B	GPT-5.2
Verification	GPT-5	GPT-5.2

D.2 CONSTRUCTION COST

Table 7 reports the per-stage construction cost of DDS, and Table 8 compares call counts with the two baselines that also construct problems. Random and MICOTA construct no new problems and therefore have no problem-construction stage to compare. DDS and EvalTree draw on the same student inference run over the 15,010 seed problems and on the same 5,664 student failures; EvalTree additionally embeds every seed problem and builds its capability tree, and TreeSynth never uses the student. The call counts are a proxy for construction burden rather than a normalized cost: DDS and EvalTree request one problem per call, whereas TreeSynth requests ten, and its 4,554 generation calls yielded 42,912 problems. The hosted models also do not log reasoning tokens, so calls cannot be converted into tokens. On teacher solution generation, the stage whose calls are most comparable across the three methods, DDS issues 20,000 calls against 19,391 for EvalTree and reaches 45.02 against 42.20 on the held-out math average. Diagnostic crossover adds no stage, since it places two diagnoses in one generation request; its cost appears as longer solutions, with 203.8M solution tokens for DDS against 154.6M for DDS-AUG.

Table 7: Per-stage construction cost of DDS.

Stage	Model	Calls	Output
Student inference	Qwen2.5-3B-Instruct (local)	15,010	15,010 solutions, 18.9M tokens
Diagnosis	GPT-5 (API)	15,010	5,664 diagnosed failures
Problem generation	GPT-5 (API)	25,000	24,985 problems
Candidate verification	GPT-5 (API)	24,985	20,324 accepted
Solution generation	QwQ-32B (local)	20,000	20,000 solutions, 203.8M tokens
Format cleaning	rule-based	–	16,191 retained
Response verification	GPT-5 (API)	16,191	15,261 accepted
Training set	–	–	12,000 examples, 98.9M tokens

Table 8: Construction calls per stage for the methods that construct new problems. *Lower bound, since empty responses are not stored.

Stage	DDS	EvalTree	TreeSynth
Student inference (local)	15,010	15,010	–
Diagnosis / capability annotation	15,010	15,010	–
Capability embedding	–	15,010	–
Tree construction	–	10,022	2,179
Problem generation	25,000	$\geq$ 19,391*	4,554
Candidate verification	24,985	–	–
Solution generation (local)	20,000	19,391	15,000
Response verification	16,191	–	–

D.3 Reduced Patch Budget

We retrain DDS with 7K patches, against 12K for every baseline, under the same fine-tuning configuration. As Table 9 shows, DDS with 7K patches remains above every 12K baseline in Table 1 and Figure 3 on both the held-out math average and seed-failure recovery.

Table 9: DDS with a reduced 7K patch budget against EvalTree and TreeSynth at 12K (Pass@1).

Method	Seed Failure	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
EvalTree (12K)	20.5	84.9	67.9	11.3	44.1	32.1	12.9	42.20
TreeSynth (12K)	20.8	86.1	67.4	8.8	43.8	33.3	12.1	41.92
DDS (7K)	27.2	87.1	69.1	8.9	45.6	32.8	15.0	43.08
DDS (12K)	28.5	87.6	71.9	10.4	46.2	37.3	16.7	45.02

D.4 Verification

Verification runs at two stages: the candidate check rejects 18.7% of generated problems, and the response check rejects 5.7% of the teacher responses it evaluates (Table 7). Composing two diagnoses makes well-posedness harder to achieve: the verifier judges 6.3% of DDS candidates ill-posed, against 2.4% for DDS-Aug. We manually inspected 10 randomly sampled flagged candidates and found all of them ill-posed. To test whether filtering explains the gains, we remove both verification stages from DDS and keep only the format cleaning that drops responses without a parseable final answer; we apply the same cleaning to EvalTree and TreeSynth and rebuild each at the same 12K budget. As Table 10 shows, each method moves only slightly and DDS keeps its margin, so the verifier's role is to keep ill-posed problems out of the patch set rather than to produce the gains.

Table 10: Removing verification. "Format cleaning only" drops responses without a parseable final answer and applies no other filter.

Method	Held-out math avg	Seed-failure recovery
Base model	37.48	15.9
EvalTree	42.20	20.5
EvalTree, format cleaning only	41.1	21.3
TreeSynth	41.92	20.8
TreeSynth, format cleaning only	42.0	22.6
DDS	45.02	28.5
DDS, format cleaning only	45.1	29.1

D.5 Decontamination

The seed set contains 15,010 problems: 5,000 each from GSM8K, MATH, and OpenMathReasoning, and 10 from AIME 2024 or earlier. The held-out suite uses the official test splits and AIME 2025, which postdates every AIME problem in the seed set. An exact-match check on whitespace-normalized problem text finds no overlap between the three benchmarks that share a source with the seed set and either the seed set or the final 12K patch set (Table 11). Near-duplicate screening with word 13-gram Jaccard similarity at a threshold of 0.80 also finds none; the highest similarity to the patch set is 0.02. A stricter unigram Jaccard audit at the same threshold flags 19 MATH problems that share a template with a seed problem; DDS does not train on seed problems, and the patch set contains no such match (highest similarity 0.43).

Table 11: Exact-match overlap between held-out benchmarks and the training data.

Benchmark	Overlap with seed set (15,010)	Overlap with patch set (12K)
GSM8K (500)	0	0
MATH (500)	0	0
AIME25 (30)	0	0

E Additional Math Results

Table 12: Pass@8 benchmark results on Qwen2.5-3B-Instruct (T=0.8, p=0.95, N=8) under a 5-epoch fine-tuning schedule.

Method	Seed Fail	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
Qwen2.5-3B-Instruct	43.6	95.2	85.6	10.0	72.5	48.6	36.7	58.10
DDS (Ours-Aug)	53.8	95.2	90.4	**26.7**	75.0	58.8	**50.0**	66.02
DDS (Ours-Evolve)	**55.2**	**96.0**	**91.0**	23.3	**82.5**	**61.2**	**50.0**	**67.33**
Random	51.0	95.0	88.2	16.7	75.0	54.4	**50.0**	63.22
MICOTA	41.8	95.0	85.2	10.0	67.5	50.2	33.3	56.87
EvalTree	47.4	95.6	87.8	**26.7**	72.5	55.3	43.3	63.53
TreeSynth	49.2	**96.0**	86.2	23.3	80.0	56.0	33.3	62.47

Table 13: Pass@8 benchmark results on Qwen2.5-3B-Instruct (T=0.8, p=0.95, N=8) under a 10-epoch fine-tuning schedule, included as a robustness check.

Method	Seed Fail	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
Qwen2.5-3B-Instruct	43.6	95.2	85.6	10.0	72.5	48.6	36.7	58.10
DDS (Ours-Evolve)	**63.6**	**95.8**	**91.2**	**26.7**	70.0	**60.9**	**46.7**	**65.22**
Random	56.8	94.6	88.0	13.3	70.0	52.7	43.3	60.32
MICOTA	37.8	93.4	79.4	13.3	65.0	47.1	20.0	53.03
EvalTree	41.0	94.0	86.6	23.3	62.5	50.8	36.7	58.98
TreeSynth	50.4	95.2	87.0	23.3	**82.5**	56.0	30.0	62.33

Table 14: Pass@8 benchmark results on Llama-3.2-1B-Instruct (T=0.8, p=0.95, N=8) under a 10-epoch fine-tuning schedule.

Method	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
Llama-3.2-1B-Instruct	13.4	43.0	0.0	**42.5**	14.7	**13.3**	21.15
DDS (Ours-Evolve)	**79.8**	**55.4**	0.0	40.0	**20.6**	**13.3**	**34.85**
Random	70.4	53.6	0.0	25.0	**20.6**	10.0	29.93
MICOTA	30.2	15.8	0.0	20.0	6.8	0.0	12.13
EvalTree	72.2	44.2	0.0	22.5	10.5	6.7	26.02
TreeSynth	67.0	47.2	0.0	32.5	16.0	6.7	28.23

F Robustness and Side-Effect Analyses

This section provides robustness and side-effect results summarized in Section 4. These analyses are not part of the main RQ1–RQ4 evidence chain, but they test whether the DDS gains are stable across random seeds and evaluation runs, robust to the teacher model, and accompanied by limited degradation on problems the student initially solved.

F.1 Run-to-Run Variance

To quantify variance from stochasticity in inference, synthesis, and training, we rerun DDS on Qwen2.5-3B-Instruct with two additional random seeds. Each run independently samples 12K crossover patches from a 20K candidate pool generated by the diagnosis and crossover stages. Seed-failure recovery is 28.48 ± 0.05 and the held-out math average is 44.78 ± 0.21, indicating that the main DDS gains are stable across independent runs.

Table 15: **DDS variance across three seeds.** Pass@1 mean $\pm$ standard deviation on Qwen2.5-3B-Instruct.

Method	Seed Failure	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
DDS (ours, mean $\pm$ std, n=3)	$\mathbf{28.48 \pm 0.05}$	87.36 ± 0.46	71.79 ± 0.30	10.83 ± 1.10	47.28 ± 1.32	36.56 ± 0.65	14.87 ± 2.15	$\mathbf{44.78 \pm 0.21}$

F.2 Teacher-Model Robustness

DDS uses a teacher LM to generate reference solutions; diagnosis, problem generation, and verification are not varied in this test. To test whether the gains depend on a particular teacher, we replace the QwQ-32B teacher used in the main experiments with Qwen3-8B and rerun DDS, EvalTree, and TreeSynth on Qwen2.5-3B-Instruct under the same 12K patch budget and fine-tuning recipe. DDS retains the largest seed-failure recovery and held-out math average.

Table 16: **Teacher-model robustness on Qwen2.5-3B-Instruct.** Replacing QwQ-32B with Qwen3-8B preserves the DDS advantage over EvalTree and TreeSynth at the same 12K patch budget.

Method	Seed Failure	GSM8K	MATH	AIME25	AMC	Olympiad	CN-Math24	Math Avg
EvalTree	14.6	85.7	63.3	6.2	36.6	27.7	7.9	37.90
TreeSynth	16.7	86.8	64.1	5.0	36.2	27.0	4.6	37.28
DDS (ours)	**29.1**	**88.1**	**71.7**	**9.6**	**45.6**	**34.3**	**15.8**	**44.18**

F.3 Multimodal Evaluation Variance

Table 3 reports means over five evaluation runs, and Table 17 gives the corresponding 95% confidence intervals. DDS has the highest mean on all seven benchmarks; its interval is disjoint from that of TreeSynth, the strongest baseline, on five of them and overlaps on MMVet (+0.2) and RealWorldQA (+0.6). The verifier rejects 12% of the generated multimodal candidates.

Table 17: Multimodal results: mean $\pm$ 95% confidence interval over five evaluation runs.

Method	OCR	MATHVISTA	MATHVISION	MMVET	HALLUSION	DYNAMATH	REALWORLD
Base model	77.6 ± 0.6	54.2 ± 1.1	22.5 ± 0.7	63.0 ± 0.8	38.9 ± 1.1	40.4 ± 1.3	52.6 ± 0.6
Random	80.2 ± 0.5	56.8 ± 1.1	23.4 ± 0.8	66.8 ± 0.9	39.4 ± 1.5	41.9 ± 1.3	53.7 ± 0.8
TreeSynth	83.2 ± 0.3	67.1 ± 0.9	27.5 ± 0.6	71.1 ± 0.3	41.5 ± 1.1	48.8 ± 1.3	64.5 ± 0.7
DDS (ours)	84.2 ± 0.3	69.5 ± 0.9	29.3 ± 0.7	71.3 ± 0.4	47.2 ± 1.2	52.9 ± 1.2	65.1 ± 0.6

F.4 SIDE EFFECTS ON INITIALLY CORRECT PROBLEMS

We examine whether fine-tuning on DDS patches induces side effects on problems the student already answered correctly. Within a 500-problem diagnostic subset, only 9 of 500 cases drop from Pass@8 ≥ 1 before fine-tuning to $0/8$ after fine-tuning. Among problems originally answered with full robustness ($8/8$), 33 lose some robustness, with post-fine-tuning scores between $3/8$ and $7/8$. The volume of side effects is small relative to the recovery gains, but not zero.

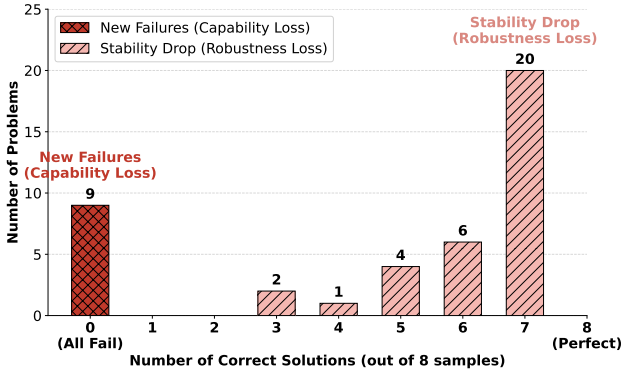


Figure 5: **Side effects of DDS fine-tuning on a 500-problem diagnostic subset.** *New failures* (x=0): problems with at least one correct seed response before fine-tuning that drop to $0/8$ after. *Stability drop* ($x \in [3, 7]$): problems originally answered with full robustness ($8/8$) that retain partial robustness.

G PATCH–IMPROVEMENT ALIGNMENT ANALYSIS

This section provides the full patch–improvement alignment analysis summarized in Section 4.4. The analysis is post-hoc and does not claim causal attribution. Instead, it asks whether the evaluation problems improved by a fine-tuned model are semantically close to examples in that model's own patch set.

Setup. For each of three evaluation sources (MATH, OlympiadBench, and the seed failures used in Section 4.2) and each fine-tuned method, we partition the improved problems into two regimes. Case 1 is strict recovery, where the base student answers $0/8$ samples correctly and the fine-tuned student answers at least one sample correctly. Case 2 is delta improvement, where the base student answers 1–$7/8$ samples correctly and the fine-tuned student strictly increases that count. For each improved problem, we form retrieval queries from the target problem and the response slots that flipped from incorrect to correct, capped at four queries per problem. We embed these queries and the entire 12K patch set with Qwen3-Embedding-8B (Zhang et al., 2025). For each training sample, we mean-aggregate similarity across the per-problem queries, producing a ranked list of training samples for every improved problem. We report the mean similarity of the top-500 training samples and summarize across improved problems within each method, source, and case group using the median, with population standard deviation as the error bar.

Findings. Three observations are consistent across sources. First, DDS drives more improvements: summing across regimes, DDS totals 190 improved problems on MATH, 276 on OlympiadBench,

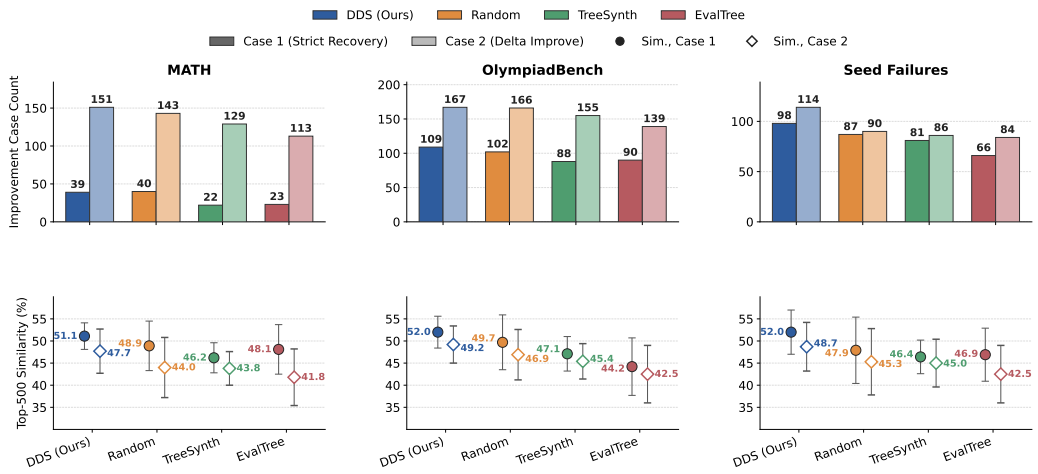


Figure 6: **Patch–improvement alignment analysis** on three evaluation sources: MATH, OlympiadBench, and seed failures. *Top*: number of improved evaluation problems per method, split by improvement regime. Dark bars show Case 1 strict recovery ($0/8 \rightarrow \geq 1/8$); light bars show Case 2 delta improvement (1–$7/8 \rightarrow$ strictly larger). *Bottom*: top-500 mean similarity between improved-problem queries and each method's 12K patch set, in cosine percent. Markers are seed-level medians; error bars are population standard deviations across improved problems within the group.

and 212 on seed failures, exceeding the best non-DDS method by 7, 8, and 35 problems respectively. Second, the improvements that DDS produces are more strongly aligned with its own patches: top-500 similarity for DDS is the highest of the four methods on every source/case cell, with Case 1 medians of 51.1, 52.0, and 52.0 on MATH, OlympiadBench, and seed failures, and Case 2 medians of 47.7, 49.2, and 48.7. The closest baseline is Random, trailing by 2.2 to 4.1 percentage points on Case 1. Third, the alignment advantage holds in both regimes: Case 1 similarities are uniformly higher than Case 2 similarities for every method, consistent with the larger per-problem behavioral change in strict recoveries.

Interpretation. The case-count panel shows that DDS is not concentrating its gains on a small set of easy problems; it improves the largest number of distinct evaluation items. The similarity panel shows that, for those improved items, the training samples most related to each improvement are systematically closer in embedding space to the improved solution behavior than the corresponding samples drawn from baseline patch sets. Because every method retrieves from a patch set of the same size (12K), a higher value cannot come from a larger retrieval pool; it requires the patches selected by trace-centric diagnosis and diagnostic crossover to land near, in representation space, the responses the student later produces correctly. This supports the interpretation that diagnosis-conditioned synthesis aligns the patch set with the repairs the student later makes.

H Full Crossover Example

This section provides the full parent problems, diagnoses, targeted syntheses, and crossover problem for the worked example summarized in Figure 2.

Table 18: Conceptual summary of a diagnosis-driven crossover example (Case 31).

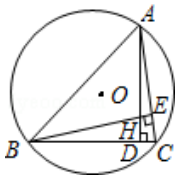
Component	Summary
Parent A	**Distinct-outcome probability.** The original problem asks for the probability that six fair dice show six distinct values. The student correctly identifies the counting structure, with 6! favorable outcomes out of 6^6 total outcomes, but fails to carry the exact probability computation through to the reduced final form.
Targeted synthesis A	DDS generates a new distinct-outcome probability problem using repeated spins of a 9-outcome spinner. This preserves the same repaired skill: correctly counting distinct outcomes and simplifying the resulting probability.
Parent B	**Geometric area reasoning.** The original problem asks for the shaded area of a tiled floor where each square tile contains four white quarter-circles. The student incorrectly treats the four quarter-circles of radius $1/2$ as having total area π, rather than recognizing that together they form one full circle of radius $1/2$, with area $\pi/4$.
Targeted synthesis B	DDS generates a new quarter-circle area problem with a different wall size and radius. This directly targets the same geometric misunderstanding: aggregating four quarter-circles into one full circle of the correct radius.
Diagnostic crossover	**A unified geometric-probability game.** The crossover creates a single game in which six dice must show distinct values and six independently dropped beads must all land in the gray region of a square board. The geometry is not a separate afterthought: the gray area directly becomes the success probability for each bead. The final probability is $\frac{5}{324}\left(1-\frac{\pi}{9}\right)^6$.

Table 19: Full Case 31 example: parent problems, teacher diagnoses, targeted syntheses, and diagnostic crossover. Parent A concerns exact probability counting, while Parent B concerns geometric area reasoning.

Component	Content
Parent A original problem	What is the probability of rolling six standard, six-sided dice and getting six distinct numbers? Express your answer as a common fraction.
Parent A teacher diagnosis	The setup is correct: total outcomes 6^6 and favorable outcomes 6! are correct. The sole issue is an arithmetic mistake when simplifying the fraction, incorrectly identifying the GCD and performing an invalid cancellation, which led to the wrong final probability.
Targeted synthesis A	Maya plays a game using a fair spinner with nine equally likely outcomes labeled 1 through 9. She spins the spinner five times. What is the probability that all five results are distinct? Express your answer as a common fraction. *Targeted A answer:* $560/2187$.
Parent B original problem	An 8-foot by 10-foot floor is tiled with square tiles of size 1 foot by 1 foot. Each tile has a pattern consisting of four white quarter circles of radius $1/2$ foot centered at each corner of the tile. The remaining portion of the tile is shaded. How many square feet of the floor are shaded?
Parent B teacher diagnosis	The student conceptually misinterpreted the geometry of the pattern: four quarter circles of radius $1/2$ combine to exactly one full circle of radius $1/2$, with area $\pi/4$. Using π for the white area per tile leads to subtracting far too much, hence $80-80\pi$. The correct shaded area per tile is $1-\pi/4$, yielding a total shaded area of $80-20\pi$.
Targeted synthesis B	A 7-foot by 9-foot mosaic wall is covered with 1-foot by 1-foot square tiles. On each tile, a clear varnish design leaves uncoated four quarter-circles of radius $1/3$ foot centered at the four corners of the tile; the rest of the tile is coated. How many square feet of the wall are coated? *Targeted B answer:* $63-7\pi$.
Crossover problem	Mira designs a game using a square board of side length 1. At each corner of the square, a white quarter-circle of radius $1/3$ is drawn inside the board; all remaining area is shaded gray. She rolls six fair six-sided dice and simultaneously drops six tiny beads independently and uniformly at random onto the face of the board. She wins if the six dice show six distinct numbers and all six beads land in the gray region (not in any white quarter-circle). What is the exact probability that she wins? Express your answer in simplest exact form.
Merged diagnosis	Merged: The student conceptually misinterpreted the geometry of the pattern: four quarter circles of radius $1/2$ combine to exactly one full circle of radius $1/2$, with area $\pi/4$. Using π for the white area per tile leads to subtracting far too much, hence $80-80\pi$. The correct shaded area per tile is $1-\pi/4$, yielding a total shaded area of $80-20\pi$. + The setup is correct: total outcomes 6^6 and favorable outcomes 6! are correct. The sole issue is an arithmetic mistake when simplifying the fraction, incorrectly identifying the GCD and performing an invalid cancellation, which led to the wrong final probability.
Crossover answer	$\frac{5}{324}\left(1-\frac{\pi}{9}\right)^6$.

I QUALITATIVE MULTIMODAL EXAMPLES

This section provides qualitative examples comparing DDS and TREESYNTH in the multimodal setting discussed in Section 4.6.



Original Seed Problem

As shown in the figure, $\triangle ABC$ is inscribed in circle O. $AD \perp BC$ at point D, and $BE \perp AC$ at point E; AD and BE intersect at point H. If $BC = 6$ and $AH = 4$, what is the radius of circle O? ()

Choices:
(A) 5
(B) $2\sqrt{13}$
(C) $\sqrt{13}$
(D) 5.5

TREESYNTH Problem

As shown in the figure, $\triangle ABC$ is inscribed in circle O. $AD \perp BC$ at point D, and $BE \perp AC$ at point E; AD and BE intersect at point H. If $BC = 8$ and $AH = 6$, what is the radius of circle O? ()

Choices:
(A) 4
(B) 5
(C) $\sqrt{41}$
(D) 6

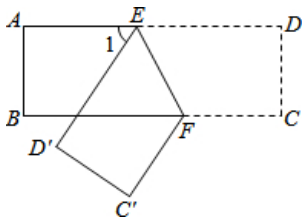
DDS Problem

Triangle ABC is inscribed in a circle of radius R. $AD \perp BC$ at D, $BE \perp AC$ at E, and AD and BE intersect at H. Given $BC = 6$ and $AH = 4$:

(a) compute $\angle A$ exactly, either by expressing it using an arctangent or by giving its exact sine and cosine; and

(b) find R using those values.

Figure 7: Qualitative comparison between a seed problem, a TREESYNTH-style generated problem, and a DDS-style generated problem. TREESYNTH preserves the same visual and geometric template while changing surface-level numerical values, whereas DDS constructs a more diagnostic patch that requires recovering the hidden angle relation before computing the circumradius.



Original Seed Problem

As shown in the figure, a rectangular sheet of paper $ABCD$ is folded along EF, such that points D and C land at positions D' and C', respectively. If $\angle 1 = 56°$, what is the measure of $\angle EFB$? ()

Choices:
(A) $56°$
(B) $62°$
(C) $68°$
(D) $124°$

TREESYNTH Problem

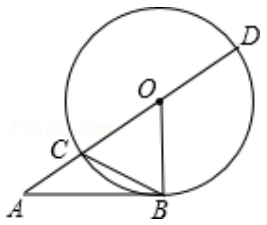
As shown in the figure, a rectangular sheet of paper $ABCD$ is folded along EF, such that points D and C land at positions D' and C', respectively. If $\angle 1 = 48°$, what is the measure of $\angle EFB$? ()

Choices:
(A) $48°$
(B) $60°$
(C) $66°$
(D) $132°$

DDS Problem

Let rectangle $ABCD$ have coordinates $A(0, 2)$, $B(0, 0)$, $C(4, 0)$, and $D(4, 2)$. Point E lies on AD at $E(3, 2)$. The paper is folded along line EF, where EF is the line through E that makes $28°$ below the horizontal, so its slope is $-\tan 28°$. Point D reflects across EF to D'.
Using this concrete setup, determine whether ED' is parallel to AB. Show all work by finding the coordinates of D' and comparing slopes.

Figure 8: Qualitative comparison between a seed problem, a TREESYNTH-style generated problem, and a DDS-style generated problem for a paper-folding geometry task. TREESYNTH preserves the original template and changes the given angle value, whereas DDS turns the folding relation into a coordinate-reflection problem that directly tests whether the student can reason about reflection geometry rather than only apply the angle-bisector shortcut.



Original Seed Problem

As shown in the figure, line AB is tangent to circle O at point B, and secant line ACD passes through the center O. If $\angle BCD = 70°$, what is the measure of $\angle A$?

Choices:
(A) $20°$
(B) $50°$
(C) $40°$
(D) $80°$

TREESYNTH Problem

As shown in the figure, line AB is tangent to circle O at point B, and secant line ACD passes through the center O. If $\angle BCD = 65°$, what is the measure of $\angle A$?

Choices:
(A) $30°$
(B) $40°$
(C) $50°$
(D) $70°$

DDS Problem

In the figure, AB is tangent to circle O at B, and secant line ACD passes through the center O. A point P lies on the circle, and chord BP is drawn. The angle between tangent AB and chord BP is $32°$, and $\angle PCD = 38°$. Use the tangent chord relation to determine the measure of arc $\overset{\frown}{BP}$, then use the fact that A, C, O, D are collinear to find $\angle BOC$ and $\angle A$. Finally, explain why treating $\angle BCD$ as an inscribed angle would give an incorrect result.

Figure 9: Qualitative comparison between a seed problem, a TREESYNTH-style generated problem, and a DDS-style generated problem for a tangent–secant circle geometry task. TREESYNTH keeps the original visual template and changes the numerical angle, while DDS creates a more diagnostic patch that requires coordinating tangent–radius perpendicularity, central-angle reasoning, and triangle angle relations.

J PROMPTS

This section lists the prompts used by DDS: the diagnostic crossover prompt (Table 20), the math failure diagnosis prompt (Table 21), and the single-failure augmentation prompt used for the DDS-AUG ablation (Table 22).

Table 20: Evolutionary Crossover Prompt

Section	Prompt Content
Role	You are an expert math exam creator. Your goal is to synthesize multiple student failure cases into a new, complex math problem that simultaneously targets the specific weaknesses identified in each case.
Input Data	Two "Parent" failure cases are provided (A and B). Each contains: 1) The original problem. 2) The student's incorrect solution (for analysis). 3) The teacher's diagnosis of the error.
Analysis Phase	Cross-reference within each parent: Analyze the Student Incorrect Solution against the Teacher Diagnosis to extract the specific failure mechanism. Do this for both Parent A and Parent B.
Generation Task	Create a SINGLE new problem that merges the mathematical challenges of Parent A and Parent B. The problem must be structured such that solving it correctly requires applying specific reasoning steps that fix BOTH identified failure mechanisms. Abstract away original surface details (names, numbers) to create a fresh scenario.
Strategy Logic	**Priority (Strategy 2):** Integrated Knowledge Fusion. Create a unified problem requiring skills from both domains simultaneously (e.g., Geometry constraints on a Calculus problem). **Fallback (Strategy 1):** Structural Dependency. Part A results are mandatory inputs for Part B. Only use Strategy 1 if Fusion is too contrived.
Constraints	Ensure the problem is well-defined: - Data Consistency (counts/types must align). - Domain Validity (probabilities $\in [0, 1]$, valid geometry). - Uniqueness (sufficient info for a single answer).
Output	Output strictly in JSON format containing: `question`, `reference_solution`, `reference_answer`, and `strategy_used`. Avoid all meta-commentary.

Table 21: Math Failure Diagnosis Prompt

Section	Prompt Content
Role	You are a STRICT math teacher analyzing a student's chain-of-thought solution. Your job is to judge correctness and, if wrong, provide a structured diagnosis.
Input Data	The model is provided with: 1) The original `Question`. 2) The normalized `Gold Answer`. 3) The `Reference Solution` (for context). 4) The `Student Solution` to analyze.
Check 1: Correctness	**Strict Binary Check**: The solution is CORRECT if and only if the final answer matches the Gold Answer after normalization (e.g., $1/2 = 0.5$, forty-two $= 42$). Ignore superficial units ("dollars") but require physical unit conversions (m vs cm). No partial credit.
Check 2: First Error	If incorrect, locate the **First Major Error**: - `first_error_span`: Quote the specific step where logic diverged. - `first_error_cause`: Explain *why* this step is wrong. - Assign local tags: `weakness_category` and `quality_level`.
Check 3: Diagnosis	If incorrect, provide a **Teacher Diagnosis**: - Give a high-level analysis of the overall failure pattern beyond just the first step. - Assign global tags: `weakness_category` and `quality_level` for the overall solution.
Taxonomy	**Weakness Categories:** Concept/Logic (structural failure), Misinterpretation (misread prompt), Calculation (arithmetic slip), ModelCollapse (nonsense/loops), or Label/DataIssue. **Quality Levels:** High (clear conceptual pattern useful for training), Medium (local slips), Low (vague/unusable).
Output	Return exactly ONE JSON object. Must include boolean `correctness`. If false, must include: `first_error_span`, `first_error_cause`, `teacher_diagnosis`, and the associated category/quality tags.

Table 22: Sequential Augmentation Prompt

Section	Prompt Content
Role	You are an expert math teacher and data annotator. Your goal is to create a NEW math problem designed to address a specific failure mode exhibited by a student.
Input Data	The model is provided with: 1) The `Original Reference Problem`. 2) The `Student's Incorrect Solution` (for analysis). 3) The `Diagnosis` (either specific First Error or global Teacher Diagnosis). 4) Pre-labeled weakness tags (Category and Quality).
History Context	*(Dynamic Section)*: "=== PREVIOUSLY GENERATED VARIANTS ===" A list of questions already generated in this session is injected here. **Constraint**: You MUST create a distinctly DIFFERENT scenario/story/numbers. Do NOT repeat the specific context of previous variants.
Task	1. Analyze the Student Solution + Diagnosis to identify the specific logic/calculation trap. 2. Create a NEW problem targeting this SAME weakness. 3. Create a fresh scenario with new numbers and names. Abstract away the original surface details, but strictly preserve the core mathematical logic or trap that caused the failure.
Style Guide	Write naturally as if showing work on paper. Show all calculations explicitly (e.g., $48 \div 2 = 24$). Avoid rigid step numbering. End with `\boxed{ANSWER}`.
Output	Respond in STRICT JSON schema containing: `question`, `reference_solution`, and `reference_answer`. Strictly avoid meta-commentary about the student, diagnosis, or reference checks.