How code helps different tasks? A decompositional lens on LLM post-training

Anonymous authors
Paper under double-blind review

Abstract

Evaluating code data as a single corpus can obscure which types of code data benefit which models and downstream tasks. Effective data selection requires understanding both the benefits of individual categories and whether these benefits persist when categories are combined. We introduce a decompositional lens for studying these effects in LLM post-training. We first decompose an execution-verified code corpus into interpretable categories based on the computational patterns of its solutions. Through controlled fine-tuning experiments, we compare individual categories with a balanced mixture across instruction-tuned models on question answering, mathematics, and code generation. The resulting response maps reveal recurring gains in average question-answering performance, while the same category can improve one model or task and degrade another. The best-performing category also varies with the starting model and target task. We then compose compact mixtures guided by these results and examine whether benefits observed in individual categories persist under joint training. On selected model–task pairs, mixtures whose constituents each improve the target task outperform both their best constituent and full-corpus training while using roughly 10–15% of the full corpus. These exploratory findings illustrate a *less is more* pattern and highlight how the value of code data in post training depends on which categories are combined for which model and task. Anonymous GitHub

1 Introduction

Code data is an appealing source of supervision for language models because its solutions express explicit computational procedures and can be checked by execution. Previous studies have investigated its benefits beyond code generation. Aryabumi et al. (2024) examine how code exposure and data quality affect downstream performance during pre-training, while Zhang et al. (2025) study how the proportion of coding data in instruction tuning affects reasoning across models and tasks. Recent work links these benefits to specific properties of the training data. Waheed et al. (2025) find that structural perturbations can affect downstream performance more strongly than semantic perturbations. Twist et al. (2026) show that restricting training data to selected structural-complexity ranges can outperform training on a broader range of complexities. These findings motivate a finer-grained understanding of which code examples provide useful supervision for a particular model and downstream task.

This understanding has practical value for data selection. Research on data mixture optimization and skill-based sampling shows that selecting and combining training data can improve downstream performance and training efficiency (Xie et al., 2023; Chen et al., 2023a). Targeted instruction tuning further demonstrates the value of selecting examples that support specific downstream capabilities (Xia et al., 2024). Within a code corpus, reference solutions express different computational patterns, such as dynamic programming, hash-based counting, and string parsing. Grouping examples by the computations performed in their solutions allows us to compare the effects of distinct forms of code supervision across models and tasks. Recombining these groups then tests whether the benefits identified in individual categories extend to joint training. Existing findings about code exposure, representation, and complexity do not by themselves specify which computational categories benefit a given model, or how their benefits change when categories are trained together. Connecting category-specific effects to joint training could therefore inform the construction of compact mixtures that improve performance with less training data.

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.

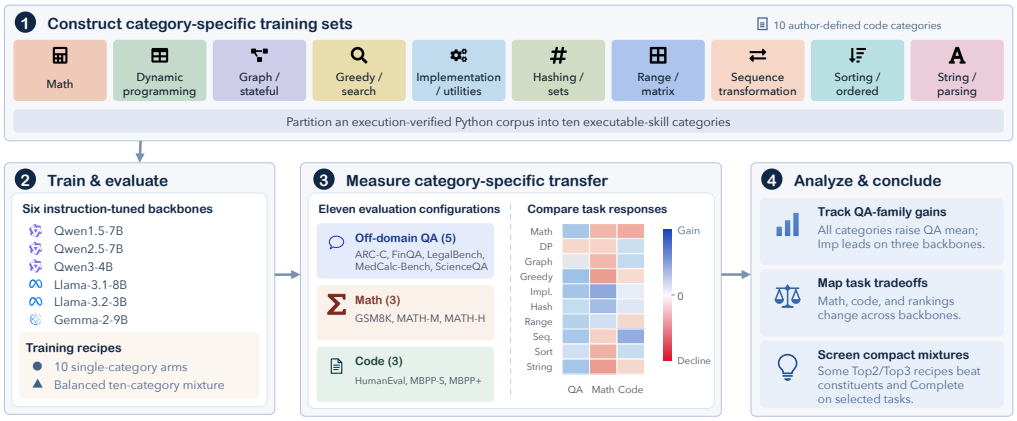


Figure 1: **Overview of the category-specific transfer study.** Ten computational categories define single-category training configurations and an example-matched balanced mixture across six instruction-tuned backbones and eleven task configurations. The exploratory screen retains only mixtures whose constituents exceed Base and whose joint model exceeds both its best constituent and COMPLETE on the same task. The heatmap is schematic.

In this paper, we study this connection through a decompositional lens. We first partition an execution-verified Python corpus derived from KodCode (Xu et al., 2025) into ten interpretable categories based on the dominant computational pattern of each solution. Across six instruction-tuned backbones and eleven evaluation configurations covering question answering, mathematics, and code generation, we evaluate single-category fine-tuning and balanced-mixture training relative to each backbone's starting checkpoint. We also compare categories with one another and with the example-matched balanced mixture to examine how their relative value varies across models and tasks. We then compose compact two- and three-category mixtures guided by these results and compare them with their individual constituents and full-corpus training. Figure 1 summarizes the workflow.

Our main results are summarized as follows:

- **RQ1: Code data supports cross-task transfer, with task-specific tradeoffs.** The balanced mixture improves average question-answering performance on all six backbones, while its effects on mathematics and code generation vary across models. Positive task-family averages can coexist with declines on individual tasks (Section 5).
- **RQ2: The best-performing category depends on both the starting model and the target task.** Category rankings can reverse when either the model or the task changes. For example, on the medium-difficulty MATH configuration, implementation/utilities outperforms dynamic programming for Llama-3.1, while their ordering reverses for Llama-3.2. For every backbone, at least one single-category model achieves a higher average score across our evaluation suite than the example-matched balanced mixture. These results show that broad category coverage alone does not ensure stronger performance (Section 6).
- **RQ3: Compact mixtures can outperform both their best constituent and full-corpus training.** On selected model–task pairs spanning question answering, mathematics, and code generation, two- and three-category mixtures outperform both references while using roughly 10–15% of the full corpus. In these cases, each constituent also improves on the starting model when trained individually. These exploratory findings illustrate a *less is more* pattern, with successful compositions depending on the starting model and target task (Section 7).

2 RELATED WORK

Code supervision. Ma et al. (2024) compare code exposure across training stages. Aryabumi et al. (2024) study code proportions and quality during pre-training, while Zhang et al. (2025) examine coding-data proportions during instruction tuning. Petty et al. (2025) find gains in mathematics and

structured compositional tasks alongside losses on other linguistic and knowledge-based tasks. These findings motivate examining which code examples benefit a given starting model and downstream task. Li et al. (2025) introduce CodeI/O, which constructs input–output prediction tasks with natural-language reasoning traces. Lin et al. (2025) propose Caco to synthesize and verify reasoning supervision through executable code. At inference time, Gao et al. (2023) and Chen et al. (2023b) delegate computation to interpreters in PAL and Program of Thoughts, respectively. Our study uses existing prompt–solution pairs for supervised post-training of instruction-tuned models; downstream QA uses no execution tools. Related studies examine how properties of code data affect transfer. Waheed et al. (2025) investigate programming-language representations and structural or semantic perturbations, while Twist et al. (2026) study structural complexity. Using influence functions, Kou et al. (2025) analyze how example difficulty, reasoning behaviors, and token-level features contribute to mathematical and code reasoning. Complementing these analyses, we directly compare single-category and joint training on groups defined by reference solutions' computational patterns, such as dynamic programming and string parsing.

Data selection and composition. Ilyas et al. (2022) introduce Datamodels to relate training-subset membership to model predictions. Xia et al. (2024) propose LESS to select instruction-tuning examples for target capabilities using gradient-based influence estimates. Liu et al. (2024) combine complexity, quality, and diversity measures in DEITA, while Lu et al. (2024) introduce InsTag to guide sampling using semantic and intention tags. Our categories describe how reference solutions compute answers, complementing instruction-level tags. For code generation, Lv et al. (2025) select fine-tuning subsets based on complexity and alignment with the source-data distribution, showing that selected subsets can outperform full-data training. At the mixture level, Xie et al. (2023) introduce DoReMi to learn pre-training domain weights using a proxy model. Liu et al. (2025) and Ye et al. (2025) predict mixture performance from smaller-scale experiments through RegMix and Data Mixing Laws, respectively. Chen et al. (2023a) use skill dependencies to guide sampling in Skill-it, while Shin et al. (2026) adapt sampling across instruction-tuning datasets through DynamixSFT. Dong et al. (2024) examine benefits and conflicts from mixing mathematical, code, and general instruction data. We study composition within a code corpus, first comparing category-specific transfer across models and tasks, then evaluating compact mixtures against their individual constituents and full-corpus training. We examine whether gains from individual categories persist or increase when those categories are trained together.

3 Study Design: From Code Categories to Compact Mixtures

We study code supervision by separating examples according to their computational patterns, measuring each category's effects across models and tasks, and recombining categories into compact mixtures. This connects overall transfer (RQ1), differences between categories (RQ2), and the effects of joint training (RQ3).

3.1 Decomposing code by computational pattern

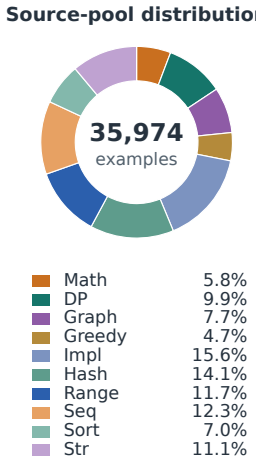
We classify each prompt–solution pair by the dominant computational pattern of its reference solution: how the solution computes the answer. This author-defined taxonomy includes algorithmic strategies, such as recurrence and greedy search, and data-processing patterns, such as hash-based counting and string parsing. Table 1 defines the ten categories through representative code patterns. Classification therefore concerns solution structure rather than the subject matter of the prompt.

The ten categories distinguish computational patterns while retaining enough examples for comparable training subsets. Each example receives one primary category, yielding ten non-overlapping groups. Auxiliary tags record additional algorithms or data structures but do not affect group assignment or training-subset selection. Computational patterns can overlap across categories, so the groups do not isolate independent reasoning abilities.

We obtain primary-category labels in two AI-assisted stages. First, an AI annotator assigns a category label and a confidence indicator. An AI reviewer then examines low-confidence cases, cases with alternative-label suggestions, and a stratified sample across categories. The reviewer sees the initial label; this AI review is not blind and does not provide independent human annotation.

Table 1: **Ten computational categories and their source-pool distribution.** Labels follow the dominant computational pattern (Section 3.1); n counts examples in each category before sampling. NT denotes number theory; impl. denotes implementation. Icons are reused in Table 4 and the response maps.

Category	Representative patterns	n
Math / NT	Modular arith. · Counting · Transforms	2,075
Dynamic prog.	Recurrence · Memoization · Tables	3,562
Graph / stateful	Traversal · Connectivity · Paths · States	2,775
Greedy / search	Greedy · Binary/local search · Rules	1,706
Impl. / utilities	Direct impl. · Bookkeeping · I/O glue	5,621
Hashing / sets	Hash maps · Counting · Sets · Dedup	5,077
Range / matrix	Windows · Prefix sums · Intervals · Grids	4,221
Sequence trans.	Map/filter · Encoding · In-place ops	4,440
Sorting / ordered	Sorting · Ranks · Comparators · Merge	2,510
String / parsing	Strings · Tokenization · Parsing · Match	3,987



3.2 Measuring category-specific transfer

For a given backbone and task, we measure transfer as the score change from the original checkpoint (Base):

$$\Delta = S_{\text{trained}} - S_{\text{Base}}. \tag{1}$$

Scores are expressed as percentages, so Δ is measured in percentage points (pp). A positive value indicates improvement, and a negative value indicates decline. For RQ1, we compare a balanced mixture of all ten categories with Base to assess transfer across task families. For RQ2, we compare individual categories with one another and with the balanced mixture, then examine how rankings change across models and tasks.

3.3 Recombining categories into compact mixtures

For RQ3, we combine selected categories into compact mixtures to test whether joint training preserves and extends the benefits observed under individual-category training. We compare each mixture with its constituents trained individually and with full-corpus training (Complete).

We focus on cases in which joint training builds on categories that already improve the target task individually. For each backbone and task, we therefore identify mixtures that meet both conditions:

1. Each constituent category, when trained individually, scores higher than Base.
2. The jointly trained mixture scores higher than both its best constituent and Complete.

The screen is exploratory: category ranking and mixture evaluation use the same evaluation suite, and qualifying cases are identified after observing the results. Appendix C.6 reports the complete screening results and associated task scores.

Section 4 specifies the models, training recipes, and evaluation protocol.

4 Experimental Setup

Backbone models. We use six instruction-tuned backbone models: Qwen2.5-7B-Instruct, Qwen3-4B-Instruct-2507, Llama-3.1-8B-Instruct, Gemma-2-9B-Instruct, Llama-3.2-3B-Instruct, and Qwen1.5-7B-Chat. All training runs start from the corresponding Base checkpoint.

Training. For RQ1 and RQ2, we apply LoRA supervised fine-tuning (Hu et al., 2022) separately to each computational category and to a balanced mixture of all ten categories. Each single-category configuration and this balanced reference use 1,706 examples and 107 optimization steps. Example counts and updates are matched across these trained configurations, but token counts are not. Appendix A provides the sampling procedures and training hyperparameters.

Compact mixtures. We rank each backbone's single-category configurations by Overall, the equally weighted mean score across the eleven evaluation configurations described below. Personalized Top2 and Top3 use the two or three highest-ranked categories, respectively. Shared Top3 uses the same ⚙ implementation/utilities (Imp), # hashing/sets, and ↓≡ sorting/ordered processing combination for all six backbones. Recipes are fixed before mixture training; their outcomes are then examined separately for each task using the criteria in Section 3.3.

Both Top3 recipes and their balanced and proportional all-category controls use 5,118 examples and 320 optimization steps. This 5,118-example balanced control is separate from the 1,706-example balanced reference used for RQ1 and RQ2. Top2 uses 3,412 examples and 214 optimization steps, with no corresponding equal-budget control. COMPLETE is a fresh LoRA fine-tuning run on the entire 35,974-example source pool. Constituent, mixture, and COMPLETE runs differ in example count and token exposure, so these comparisons do not isolate gains caused by combining categories. Appendix C provides the mixture budgets, sampling procedures, and training settings.

Evaluation. We evaluate eleven configurations spanning code generation, mathematics, and question answering (QA). Code generation includes HumanEval (Chen et al., 2021a), MBPP-Simple, and MBPP+ (Austin et al., 2021; Liu et al., 2023). Mathematics includes GSM8K (Cobbe et al., 2021) and the medium- and high-level MATH configurations (MATH-M and MATH-H) (Hendrycks et al., 2021). QA includes ARC-Challenge (Clark et al., 2018), FinQA (Chen et al., 2021b), LegalBench (Guha et al., 2023), MedCalc-Bench (Khandekar et al., 2024), and ScienceQA (Lu et al., 2022). We use pass@1 for code generation and accuracy for the remaining tasks. We report three-seed mean scores and their base-relative changes as defined in Equation 1. Task-family averages weight their constituent configurations equally; Overall uses the eleven-configuration average defined above. Table 4 groups the five QA configurations separately from the six code and mathematics configurations; task-level analyses further separate code and mathematics. Comparisons of these means are descriptive; we do not perform significance tests. Appendix A provides the evaluation settings and record availability, while Appendix B presents the complete single-category and balanced-mixture results.

5 RQ1: DOES CODE POST-TRAINING IMPROVE PERFORMANCE ACROSS TASKS AND MODELS?

We compare each backbone's starting checkpoint (Base) with the same model after balanced-mixture fine-tuning on all ten code categories. This comparison tests whether code-only post-training transfers across task families.

> **Answer.** The balanced mixture improves the QA mean on all six models, but code and mathematics gains depend on the backbone. Positive family means also conceal task-level declines.

QA improves across all six backbones. The balanced mixture raises the five-task QA mean by $+0.6$ to $+2.9$ pp (Figure 2). It improves 26 of the 30 model–task scores, including ARC-Challenge and ScienceQA on every backbone. The four declines occur on Qwen3 FinQA, Llama-3.1 LegalBench, and Qwen1.5 LegalBench and MedCalc. Thus, the positive QA mean reflects broad transfer while retaining exceptions that an aggregate alone would hide.

Code and mathematics show different model-dependent responses. The balanced mixture improves the Code mean on three backbones and the Math mean on four, including Qwen1.5's small $+0.1$ pp Math gain (Figure 3). Qwen3 gains $+1.4$ pp in Math while losing 6.5 pp in Code. Gemma-2 instead gains $+0.5$ pp in Code while losing 3.0 pp in Math. Llama-3.2 declines in both families, despite its positive QA mean. Separating Code and Math therefore exposes responses that their combined mean would obscure.

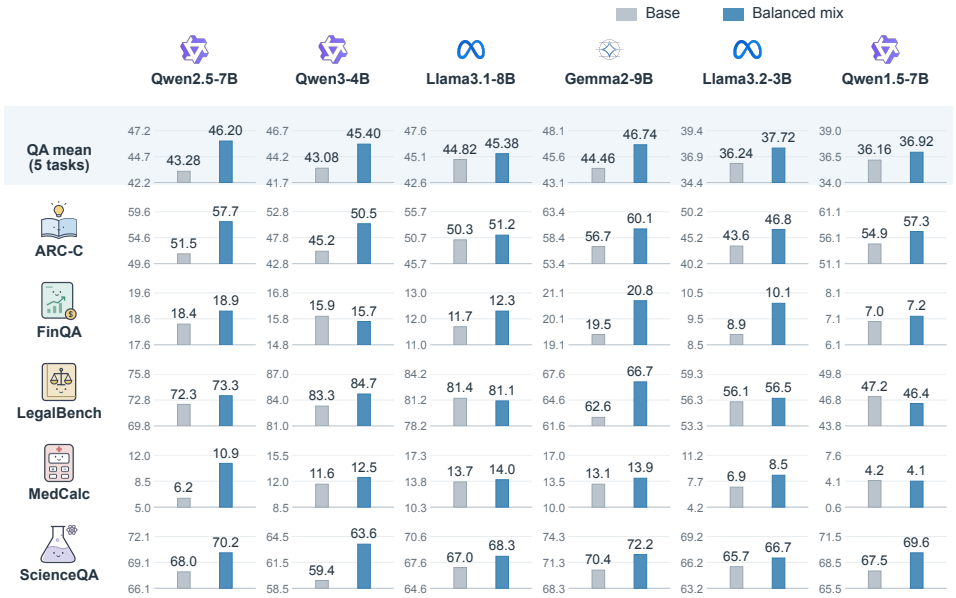


Figure 2: **QA gains are consistent across models but uneven across tasks.** Bars compare Base and Balanced mix (three-seed means, %) for the five-task QA mean and each task. QA means use equal weights and released one-decimal task scores. Each row uses the same numerical span across models, with panel-specific truncated baselines and absolute-score ticks.

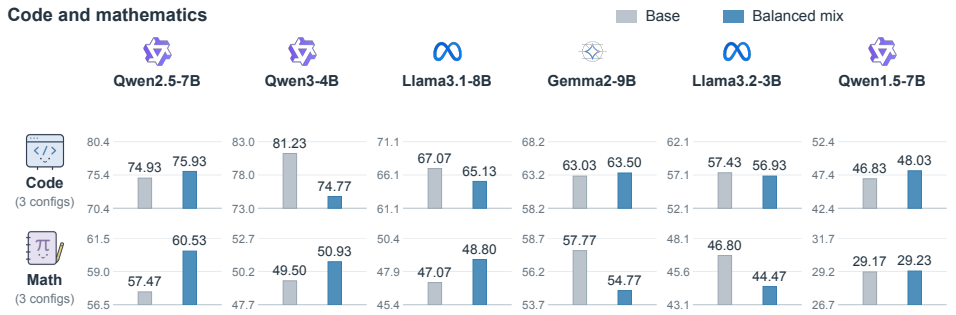


Figure 3: **Code and mathematics gains depend on the backbone.** Bars average HumanEval, MBPP+, and MBPP-Simple (Code), or GSM8K, MATH-M, and MATH-H (Math), with equal weights. Reporting conventions follow Figure 2.

QA gains extend across individual code categories. The sixty single-category training configurations provide complementary context in Appendix Table 4. Using their reported base-relative deltas, all sixty improve the QA mean and 44 improve Overall. Only 24 improve the combined code-and-mathematics mean, reinforcing the uneven transfer pattern. A separate single-seed audit checks how the positive QA means depend on task composition (Appendix E). All sixty configurations retain positive five-task QA means in this audit, but only 45 remain positive after excluding ARC-Challenge and ScienceQA. QA mean gains therefore extend across categories and backbones, although their distribution across individual QA tasks remains uneven.

6 RQ2: Which code categories help which tasks on which models?

We now compare code categories with one another while holding the backbone, target task, example count, and optimization steps fixed. Figure 4 shows all $6 \times 10 \times 11 = 660$ category–task responses, allowing both task-dependent and model-dependent category preferences to be examined.

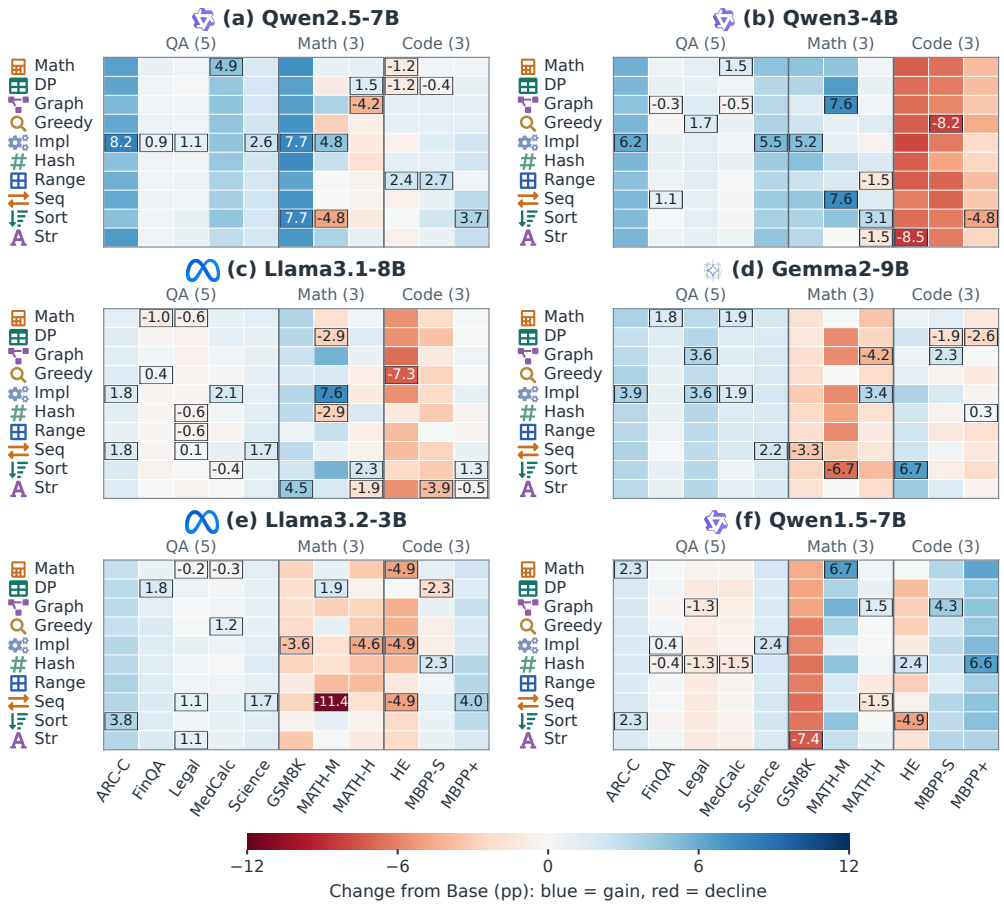


Figure 4: **Backbone-level category–task responses.** All $6 \times 10 \times 11 = 660$ cells show three-seed mean changes from the corresponding Base (pp). Rows are categories; columns group QA, math, and code tasks. Comparing rows within a task reveals category preferences; comparing panels reveals their dependence on the backbone. Blue/red denotes gain/decline on a shared scale. Outlined numbers mark the largest positive and negative response per task within each backbone, retaining ties; these are descriptive extrema. Appendix B gives every value and the balanced-mixture rows.

Answer. A code category's value depends on both the target task and the starting model. Recurring strengths exist, but no category is the preferred choice across all objectives and backbones.

Changing the target task can reverse the category ranking. On Llama-3.1, ⚙ implementation (Imp) improves MATH-M by $+7.6$ pp, whereas ⊞ dynamic programming lowers it by 2.9 pp. On MATH-H, ⚙ Imp instead changes the score by -1.1 pp and ⊞ dynamic programming improves it by $+1.5$ pp. The category ordering therefore reverses between these two mathematics configurations. A category's downstream value depends on target tasks, even within the same task family.

Changing the backbone model also changes the preferred category. Holding MATH-M fixed, the same ⚙ Imp–⊞ dynamic-programming comparison reverses between the two Llama backbones. ⚙ Imp leads on Llama-3.1, but on Llama-3.2 it lowers the score by 1.9 pp while ⊞ dynamic programming improves it by $+1.9$ pp. These reversals show that the same target task can favor different categories on different starting models.

Aggregate rankings also depend on the backbone. For example, ⚙ Imp leads Overall on Qwen2.5, whereas ⇄ sequence transformations leads on Qwen3 and ↓≡ sorting/ordered processing

on Llama-3.2. Each backbone has a single-category Overall score above its example-matched balanced mixture. The largest margin per backbone ranges from $+0.5$ to $+1.2$ pp (Appendix B.2). Thus, balanced sampling across all ten categories does not yield the highest observed Overall score under this example budget.

A recurring strength still has task-specific limits. Imp raises the QA-family mean on all six backbones and gives the largest single-category QA gain on Qwen2.5, Qwen3, and Llama-3.1. Its respective gains on these three models are 3.5, 2.9, and 0.9 pp. Yet Imp lowers FinQA and LegalBench by 0.2 pp each on Llama-3.1, and LegalBench and MedCalc by 1.1 and 0.5 pp on Qwen1.5. A recurring advantage in the QA mean therefore coexists with declines on specific tasks.

7 RQ3: Can compact mixtures outperform single-category and full-corpus training?

We next examine joint training of the Overall-ranked personalized recipes and the shared Top3 recipe defined in Section 4. After training, we compare their outcomes separately on each evaluation configuration, using the individual constituents and full-corpus training (Complete) as references.

Answer. Compact mixtures can outperform both their best constituent and full-corpus training while using only 9.5–14.2% of the examples. These gains occur on selected model–task pairs whose constituent categories each improve on Base.

Coverage of the task-level screen. We apply the conditions in Section 3.3 to the trained recipes. Each comparison corresponds to one mixture, one backbone, and one evaluation configuration. The screen retains **9 of 66 two-category comparisons** and **18 of 132 three-category comparisons**. The denominators reflect one Top2 recipe and two Top3 recipes per backbone, each evaluated on eleven configurations across six backbones. Together, the 27 qualifying comparisons span five backbones and nine evaluation configurations. Figure 5 illustrates one qualifying recipe of each size; Appendix C.6 reports the complete screen.

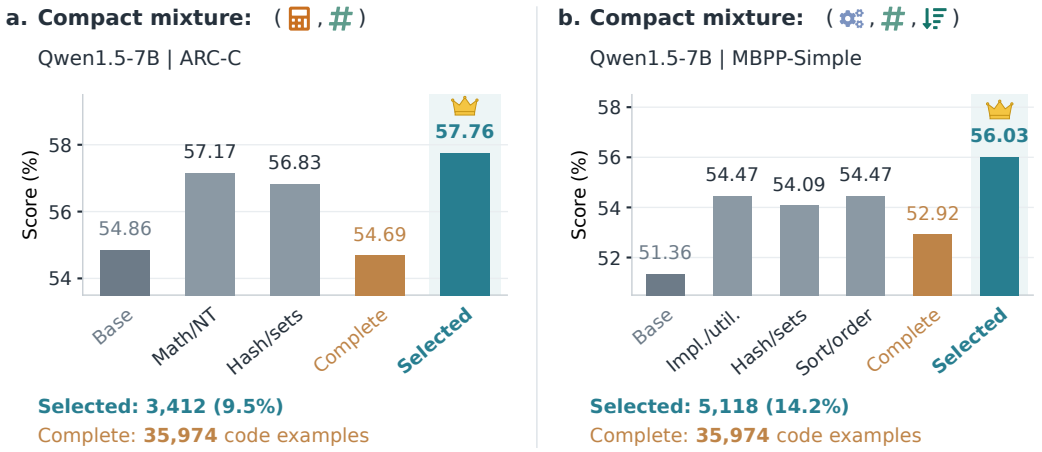


Figure 5: **Two qualifying compact mixtures on Qwen1.5.** Panel (a) shows Math + Hash on ARC-C; panel (b) shows Imp + Hash + Sort on MBPP-Simple. Selected denotes the jointly trained mixture, and Complete denotes training on all 35,974 examples. Bars show the means on truncated y axes; yellow crowns mark the highest mean. Counts and percentages give training-set sizes and fractions of Complete. See Appendix C.6 for all screened examples.

Representative gains over both references. On Qwen1.5 ARC-Challenge, the Math/Hash Top2 mixture reaches 57.76%, exceeding its best constituent by 0.60 pp and Complete by 3.07 pp. It uses 3,412 examples, or 9.5% of the full corpus. On the same backbone's MBPP-Simple, the shared Imp/Hash/Sort Top3 reaches 56.03%, exceeding its best constituent by 1.56 pp and

COMPLETE by 3.11 pp. It uses 5,118 examples, or 14.2% of the full corpus. Every constituent in both examples individually outperforms Base; Figure 5 shows the corresponding absolute scores.

Successful combinations vary across models and tasks. Even for the same target task, different model-specific recipes satisfy the screen. On ARC-Challenge, the qualifying Top2 recipe combines Imp/Str on Qwen2.5 and Sort/Imp on Llama-3.1. These examples reflect task-level outcomes of recipes selected by Overall ranking, rather than separate category selection for ARC-Challenge. The matched-budget Top3 comparisons also show that combining individually high-ranked categories does not consistently outperform balanced sampling on Overall (Appendix C.2). Individual-category rankings therefore provide candidates for joint training, but do not directly determine which mixtures will perform best. Appendix C gives the full Top3 task maps, matched-budget controls, and generation-budget sensitivity.

8 DISCUSSION

Category selection and mixture evaluation answer different questions. Category-level evaluation can narrow the candidate training pool, but single-category rankings do not determine performance after joint training. The compact-mixture results and matched-budget comparisons show why both stages matter when selecting code data. Likewise, improvements in Overall or task-family means do not establish gains on a particular target task. Code data selection therefore benefits from examining both the composition of the training set and the distribution of gains across the intended tasks.

The evidence is specific to Python and the present taxonomy. Our training pool contains execution-verified Python solutions, so the observed category rankings and task responses may not extend to other languages. The ten-category taxonomy records each solution's dominant computational pattern. Broad groups such as implementation/utilities may contain finer distinctions that this analysis does not resolve. We therefore interpret category responses as effects of the defined training subsets, rather than effects of isolated reasoning abilities.

The mixture screen establishes selected cases, not a general selection rule. The space of possible mixtures extends beyond the recipes evaluated here. We tested one Personalized Top2, one Personalized Top3, and one Shared Top3 recipe per backbone (Appendix C). Personalized recipes use Overall rankings from the same evaluation suite, and qualifying task-level cases are identified after training. Moreover, constituent, mixture, and COMPLETE runs differ in training exposure. The observed advantages therefore establish neither an optimal composition nor causal synergy between categories. Independent evaluation of frozen recipes against matched-budget controls would be needed to assess a generalizable selection rule.

The mechanisms behind category-dependent transfer remain unresolved. The single-seed audit of Imp contains examples compatible with improved application of rules and stated relationships (Appendix D). However, QA gains and reversals in mathematics and code may also reflect changes in output behavior or evaluation interfaces. The item-level audits in Appendices E and D do not establish a causal mechanism or quantify variability across training seeds. They identify candidate explanations for the observed transfer patterns, whose causal contributions remain to be tested.

9 CONCLUSION

Our work introduces a decompositional lens for studying how computational categories of code data affect LLM post-training. By examining categories individually and in combination, we connect the analysis of transfer with the construction of compact training mixtures. Our experiments reveal recurring gains in average question-answering performance, alongside task-level tradeoffs and category preferences that vary across models and evaluation objectives. These findings highlight the importance of understanding which categories benefit the particular model and task being considered.

The mixture experiments further demonstrate the potential of category composition. On selected model–task pairs, compact mixtures of individually beneficial categories outperform both their best constituent and full-corpus training while using roughly 10–15% of the full corpus. These exploratory findings illustrate a *less is more* pattern and motivate strategies for selecting and combining code data that account for both category-specific transfer and the effects of joint training.

Ethics Statement

This work studies code-data selection using an existing Python corpus and established benchmarks. Improvements on these benchmarks do not establish the safety, fairness, or reliability of model outputs in deployment. In particular, results on legal, medical, and financial tasks should not be interpreted as validation for professional decision-making. AI-assisted category labels may contain errors or systematic biases, and category-level results should be interpreted with this limitation. Any reuse or redistribution of datasets, models, or derived materials should respect the applicable licenses and usage conditions. Deployment would require separate evaluation appropriate to the intended domain and users.

Reproducibility Statement

The accompanying data record preserves the reported six-model scores at their released precision. It includes absolute task scores, base-relative changes, and a long-form CSV linking each score to its model, training arm, metric, and seed identifiers. Appendix A describes the sampling, training, evaluation, and record coverage. Appendix B displays the full task-response matrices, and Appendix C documents the compact-mixture experiment, full scores, and future independent confirmation. Appendices D and E document the separate single-seed audits of Imp transfer and RQ1; their summary tables, labels, and selected case records accompany the source.

AI Use Statement

AI tools were used for data annotation, as described in the methodology and relevant appendices, and for manuscript preparation. Manuscript assistance included restructuring, language editing, numerical transcription and consistency checks, and reference checking. We take full responsibility for the final content of this paper, including the accuracy of its text, data, figures, references, claims, and interpretations.

References

Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. To code, or not to code? Exploring impact of code in pre-training. *arXiv preprint arXiv:2408.10914*, 2024. URL https://arxiv.org/abs/2408.10914.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! A data-driven skills framework for understanding and training language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 36000–36040. Curran Associates, Inc., 2023a. doi: 10.52202/075280-1562. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/70b8505ac79e3e131756f793cd80eb8d-Abstract-Conference.html.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023b. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711. Association for Computational Linguistics, 2021b. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How Abilities in Large Language Models are Affected by Supervised Fine-tuning Data Composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 177–198, 2024. doi: 10.18653/v1/2024.acl-long.12. URL https://aclanthology.org/2024.acl-long.12/.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44123–44279. Curran Associates, Inc., 2023. doi: 10.52202/075280-1915. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/89e44582fd28ddfea1ea4dcb0ebbf4b0-Abstract-Datasets_and_Benchmarks.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587. PMLR, 2022. URL https://proceedings.mlr.press/v162/ilyas22a.html.

Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina S Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad W Safranek, Abid A Anwar, Andrew Zhang, Aidan Gilson, Maxwell B Singer, Amisha Dave, Andrew Taylor, Aidong Zhang, Qingyu Chen, and Zhiyong Lu. MedCalc-Bench: Evaluating large language models for medical calculations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 84730–84745. Curran Associates, Inc., 2024. doi: 10.52202/079017-2690. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/99e81750f3fdfcaf9613db2dbf4bd623-Paper-Datasets_and_Benchmarks_Track.pdf.

Siqi Kou, Qingyuan Tian, Hanwen Xu, Zihao Zeng, and Zhijie Deng. Which Data Attributes Stimulate Math and Code Reasoning? An Investigation via Influence Functions. In *Advances in Neural Information Processing Systems*, volume 38, pp. 12302–12324. Curran Associates, Inc., 2025. doi: 10.52202/085713-0412. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/12323a9ca6eb70fdc993efa13c207f88-Abstract-Conference.html.

Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. CodeIO: Condensing Reasoning Patterns via Code Input-Output Prediction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34471–34489. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25t.html.

Honglin Lin, Qizhi Pei, Zhuoshi Pan, Yu Li, Xin Gao, Juntao Li, Conghui He, and Lijun Wu. Scaling Code-Assisted Chain-of-Thoughts and Instructions for Model Reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 35204–35237. Curran Associates, Inc., 2025. doi: 10.52202/085713-1182. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3267aa172c31ae3641c32ecc6e42e5cd-Abstract-Conference.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. RegMix: Data Mixture as Regression for Language Model Pre-training. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5f67d864aae6115374fed7beddd119e0-Abstract-Conference.html.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What Makes Good Data for Alignment? A Comprehensive Study of Automatic Data Selection in Instruction Tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6091f2bb355e960600f62566ac0e2862-Abstract-Conference.html.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. #InsTag: Instruction Tagging for Analyzing Supervised Fine-tuning of Large Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9dae2a90bae49dc874ce1ca8fcc20879-Abstract-Conference.html.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2507–2521. Curran Associates, Inc., 2022. doi: 10.52202/068431-0182. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/11332b6b6cf4485b84afadb1352d3a9a-Paper-Conference.pdf.

Weijie Lv, Xuan Xia, and Sheng-Jun Huang. Data-efficient LLM Fine-tuning for Code Generation. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2025. doi: 10.1109/IJCNN64981.2025.11227906. URL https://doi.org/10.1109/IJCNN64981.2025.11227906.

Yingwei Ma, Yue Liu, Yue Yu, Yuanliang Zhang, Yu Jiang, Changjian Wang, and Shanshan Li. At Which Training Stage Does Code Data Help LLMs Reasoning? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9c2aa1e456ea543997f6927295196381-Abstract-Conference.html.

Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. How Does Code Pretraining Affect Language Model Task Performance? *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=pxxmUKKgel.

Haebin Shin, Lei Ji, Xiao Liu, Zhiwei Yu, Hyunwoo Yoo, Qi Chen, and Yeyun Gong. DynamixSFT: Dynamic Mixture Optimization of Instruction Tuning Collections. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39590–39603, 2026. doi: 10.18653/v1/2026.findings-acl.1972. URL https://aclanthology.org/2026.findings-acl.1972/.

Lukas Twist, Shu Yang, Hanqi Yan, Jingzhi Gong, Di Wang, Helen Yannakoudakis, and Jie M Zhang. Not all code is equal: A data-centric study of code complexity and llm reasoning. *arXiv preprint arXiv:2601.21894*, 2026. URL https://arxiv.org/abs/2601.21894.

Abdul Waheed, Zhen Wu, Carolyn Rosé, and Daphne Ippolito. On code-induced reasoning in LLMs. *arXiv preprint arXiv:2509.21499*, 2025. URL https://arxiv.org/abs/2509.21499.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 2024. URL https://proceedings.mlr.press/v235/xia24c.html.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, pp. 69798–69818, 2023. doi: 10.52202/075280-3059. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6980–7008. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.365. URL https://aclanthology.org/2025.findings-acl.365/.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data Mixing Laws: Optimizing Data Mixtures by Predicting Language Modeling Performance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cc84bfabe6389d8883fc2071c848f62a-Abstract-Conference.html.

Xinlu Zhang, Zhiyu Zoey Chen, Xi Ye, Xianjun Yang, Lichang Chen, William Yang Wang, and Linda Ruth Petzold. Unveiling the impact of coding data instruction fine-tuning on large language models reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25949–25957, 2025. doi: 10.1609/aaai.v39i24.34789. URL https://ojs.aaai.org/index.php/AAAI/article/view/34789.

A SIX-MODEL RESULT RECORDS AND TRAINING PROTOCOL

The six-model score record covers twelve arms and eleven evaluation configurations. Results summarize seeds 20260729, 20260730, and 20260731 and include Overall alongside the eleven task scores. The record contains 864 absolute values and 864 base-relative changes, including 792 task means and 72 Overall means. For each model, the ten single-category training configurations are separate from the `balanced_mix_10` reference and the base checkpoint.

The score and change records retain the precision exposed by the result plots. Values were transcribed from those plots and checked independently using OCR and arithmetic. Absolute scores minus the corresponding base and the independently printed changes agree within 0.1 pp; Overall also agrees with the mean of the eleven task scores within 0.1 pp. Such differences arise from separate rounding. Zero changes, including negative zero, are retained in the CSV and do not establish exact equality.

The accompanying CSVs separate these reported values from derived summaries. `absolute_scores.csv` and `delta_vs_base.csv` each contain 72 model–arm rows. `all_results_long.csv` contains 864 model–arm–metric rows with both values, seed identifiers, and source-plot filenames. Family means in Table 4 are derived from the reported task means; task-response figures use the reported deltas. These operations do not recover hidden decimal precision or per-seed variance.

A.1 SAMPLING AND TRAINING CONFIGURATION

Sampling balances example counts within the shared primary-label partition. Each single-category training configuration contains 1,706 examples, the size of the smallest category, 🔍 greedy search and optimization, sampled without replacement from the 35,974-example cleaned KodCode pool. The balanced arm draws from these single-category subsets: six categories contribute 171 examples each and four contribute 170, again without replacement, for $6 \times 171 + 4 \times 170 = 1{,}706$ examples. Auxiliary labels are not used to reselect examples. One epoch gives 107 optimizer steps under the recorded batch settings; input and supervised-token totals are not matched.

Training uses native chat templates, and evaluation uses the final adapter. Table 2 records the sampling, optimization, and implementation settings. Four GPUs schedule independent jobs rather than one four-GPU data-parallel run.

Table 2: **Documented sampling and training configuration.** Every trained arm has the same example count and update count; token totals can differ. Base receives no additional training.

Setting	Recorded value
Source and labels	35,974 cleaned KodCode training examples; selection by primary category only.
Single-category configurations	1,706 examples each; sampling without replacement.
Balanced mixture	1,706 examples drawn from the single-category subsets: six quotas of 171 and four of 170.
Sampling seed	20260730; no auxiliary-label reselection.
Input and loss	Native model chat template; assistant-only loss.
LoRA	Rank 16, scaling parameter 32, dropout 0.05; attention query, key, value, and output projections.
Learning rate	2×10^{-5}.
Epochs and steps	One epoch; 107 optimizer updates per trained arm.
Batch size	One example per GPU; gradient accumulation 16; effective batch size 16.
Schedule	Cosine; warmup ratio 0.03; weight decay 0.01.
Precision and length	BF16; maximum sequence length 4,096.
Checkpoint choice	Final adapter; no selection by test score.
Hardware	Four H100 80GB GPUs running independent single-GPU jobs, not four-GPU DDP.
Software	PyTorch 2.7.1, Transformers 5.10.2, PEFT 0.19.1; AOSS-restored environment.

A.2 Evaluation configuration and record coverage

Evaluation uses task-specific prompts, answer extraction, and scoring implementations. Code generation uses pass@1. ARC-Challenge, LegalBench, and ScienceQA use option log-probability scoring; the other configurations use generation. Table 3 lists the effective scoring counts and generation limits. The main analysis aggregates saved scores without regenerating answers. Appendix D checks the saved ARC-Challenge and ScienceQA predictions against their option scores and gold answers. Appendix E extends the single-seed audit to QA composition, mathematical answer extraction, and code execution records. Both retain the original scores; the alternative GSM8K extraction rule is reported separately as a sensitivity analysis.

Table 3: **Documented evaluation configuration.** Counts are the effective scored problems for each configuration. A dash indicates choice scoring, with no generation limit applicable.

Configuration	Scored n	Metric	Scoring mode	Max. tokens
ARC-Challenge	1,172	Accuracy	Option log probability	—
FinQA	1,133	Accuracy	Generated answer	128
HumanEval	164	Pass@1	Generated code	2,048
LegalBench	1,689	Accuracy	Option log probability	—
MATH-M	105	Accuracy	Generated answer	512
MBPP+	378	Pass@1	Generated code	2,048
MedCalc	1,100	Accuracy	Generated answer	128
MATH-H	262	Accuracy	Generated answer	512
GSM8K	1,319	Accuracy	Generated answer	256
MBPP-Simple	257	Pass@1	Generated code	2,048
ScienceQA	2,224	Accuracy	Option log probability	—

Generation and choice scoring use separate batch sizes. Decoding settings are fixed: generation uses temperature 0.2, top-p 0.95, and batch size four; choice scoring uses batch size 24. The Llama-3.2 template date is fixed to 22 Sep 2026. The environment does not install the optional `math-verify` package, so mathematical answers use the evaluator's fallback matching logic and retain its parser compatibility limits.

Overall weights the eleven task percentages equally while retaining their distinct scoring populations. The source field `overall_mean_no_health` is the arithmetic mean of the eleven task columns; HealthBench, APPS, and PlanBench are excluded. FinQA starts with 1,147 records and skips fourteen without target answers, leaving 1,133 scored records. MBPP-Simple and MBPP+ may overlap, so their counts cannot be summed as a disjoint MBPP test set. Upstream score CSVs are documented at ten decimal places, while the result plots expose the one-decimal values retained in this paper's extracted CSVs.

The additional audit record supplies unrounded scores, saved QA predictions, training-file hashes, and token-budget records for seed 20260730 (Appendix D). Exact replication of the reported three-seed experiment still requires corresponding records for seeds 20260729 and 20260731, including confirmation of sampling and hyperparameters, mixture quotas, optimizer implementation, checkpoint/tokenizer/dataset revisions, token exposure, and unrounded scores. The single-seed audit does not fill these gaps or estimate seed variability.

B AGGREGATE AND COMPLETE TASK-RESPONSE MAPS

B.1 TASK-FAMILY SCORES FOR ALL TRAINING RECIPES

Table 4 retains the complete task-family comparison for Base, the balanced mixture, and all ten single-category recipes. Figures 2 and 3 in Section 5 focus on Base versus the balanced mixture and separate the code and mathematics means.

Table 4: **Code post-training transfers across tasks, with uneven gains across models.** Entries are three-seed mean task-group scores (%), averaged from the reported one-decimal task scores. All 60 single-category training configurations improve the QA mean (a), whereas only 24 improve the combined code-and-mathematics mean (b). Blue/red shading indicates higher/lower performance relative to the same backbone's Base, using averages of the reported base-relative deltas. Panel (a) saturates at $+3.5$ pp and panel (b) at ± 3 pp. Base rows are unshaded; the balanced mix matches each single-category configuration's example budget.

(a) QA (5 configurations)

Training recipe	Qwen2.5-7B	Qwen3-4B	Llama3.1-8B	Gemma2-9B	Llama3.2-3B	Qwen1.5-7B
Base	43.3	43.1	44.8	44.5	36.2	36.2
Balanced mix	46.2	45.4	45.4	46.7	37.7	36.9
Math / NT	46.2	45.9	45.2	47.0	37.2	36.9
Dynamic programming	46.0	45.4	45.4	46.7	37.6	36.4
Graph / stateful	45.9	44.9	45.3	46.8	37.5	36.5
Greedy / search	46.2	45.5	45.4	46.6	37.6	36.6
Impl. / utilities	46.8	46.0	45.8	47.0	37.8	36.7
Hashing / sets	45.8	45.7	45.3	46.7	37.6	36.3
Range / matrix	45.9	45.2	45.1	46.6	37.8	36.6
Sequence transform.	46.1	45.5	45.7	46.6	38.1	36.7
Sorting / ordered	45.9	45.2	45.2	46.6	37.9	36.7
String / parsing	45.9	45.6	45.3	46.6	38.0	36.6

(b) Code & math (6 configurations)

Training recipe	Qwen2.5-7B	Qwen3-4B	Llama3.1-8B	Gemma2-9B	Llama3.2-3B	Qwen1.5-7B
Base	66.2	65.4	57.1	60.4	52.1	38.0
Balanced mix	68.2	62.9	57.0	59.1	50.7	38.6
Math / NT	67.7	64.1	56.1	59.6	51.1	40.1
Dynamic programming	67.3	64.4	55.9	58.4	51.9	38.5
Graph / stateful	67.9	64.3	57.3	58.9	50.7	39.4
Greedy / search	67.4	62.9	55.9	59.6	50.8	38.2
Impl. / utilities	68.7	63.6	57.6	60.4	49.8	38.4
Hashing / sets	68.4	64.6	56.7	59.3	51.1	39.7
Range / matrix	68.1	62.9	56.9	58.8	51.1	37.5
Sequence transform.	67.9	65.0	56.8	59.4	49.5	38.2
Sorting / ordered	67.5	64.3	58.1	58.9	52.4	38.3
String / parsing	68.7	62.8	56.2	60.6	51.4	38.3

B.2 AGGREGATE RESPONSES ACROSS BACKBONES

Single-category training improves Overall in 44 of the 60 model–category combinations. Figure 6 reports changes from each model's base checkpoint: thirteen combinations decrease and three round to zero. The distribution differs substantially across backbones. All ten categories improve Qwen2.5-7B, with gains of $+1.8$ to $+2.9$ pp. Nine improve Gemma-2-9B and Qwen1.5-7B, seven improve

Qwen3-4B, five improve Llama-3.2-3B, and four improve Llama-3.1-8B. These counts describe the directions of the reported means.

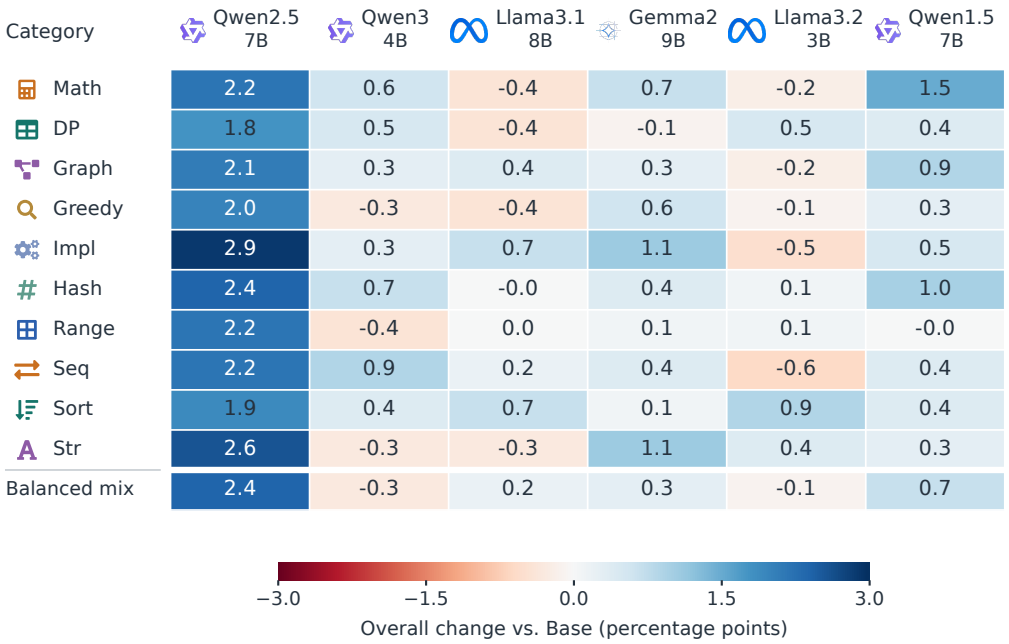


Figure 6: **Overall responses depend on both the backbone and the training category.** Cells show the reported three-seed mean Overall change from the corresponding base checkpoint, in percentage points, for ten single-category training configurations and the balanced ten-category mixture. Blue denotes improvement and red denotes decline, with one shared scale. Category icons follow Table 1. Values are reported to one decimal place.

The magnitude of these changes also differs across backbones: the strongest aggregate gains occur on Qwen2.5, while the Llama models show smaller changes of both signs. Direct implementation raises Qwen2.5 Overall from 55.8% to 58.7%. By comparison, category responses span -0.4 to $+0.7$ pp on Llama-3.1 and -0.6 to $+0.9$ pp on Llama-3.2. An aggregate close to base can nevertheless combine sizable opposing task changes. Table 4 separates QA from code and mathematics, and Figure 4 resolves each task. These views identify which task responses contribute to the aggregate change.

B.3 Complete per-backbone task responses

To expose the task changes underlying these aggregates, the per-backbone maps retain all category–task responses and the balanced-mixture reference. Every cell is a reported three-seed change from that backbone's base checkpoint in percentage points, shown at the source's one-decimal precision. All maps use the same zero-centered scale, with blue denoting improvement and red denoting decline. Base-relative deltas are plotted directly rather than reconstructed from rounded absolute scores. Category icons match Table 1; the final row is the balanced ten-category mixture. The companion CSVs retain all absolute scores as well as these changes.

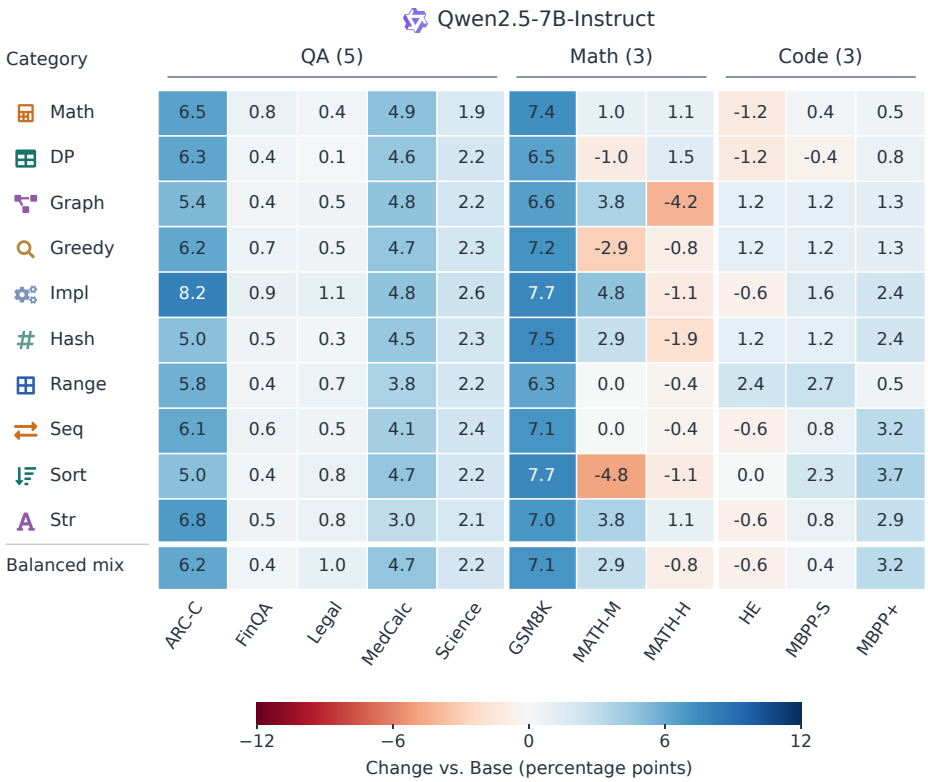


Figure 7: **Qwen2.5-7B-Instruct: complete task responses.**

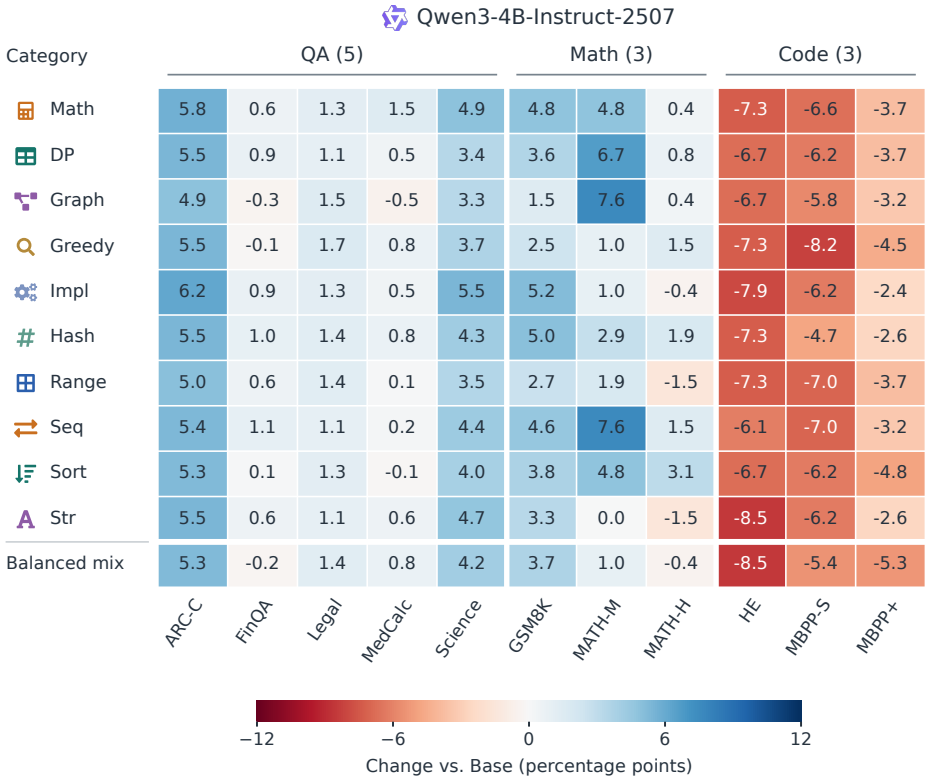


Figure 8: **Qwen3-4B-Instruct-2507: complete task responses.**

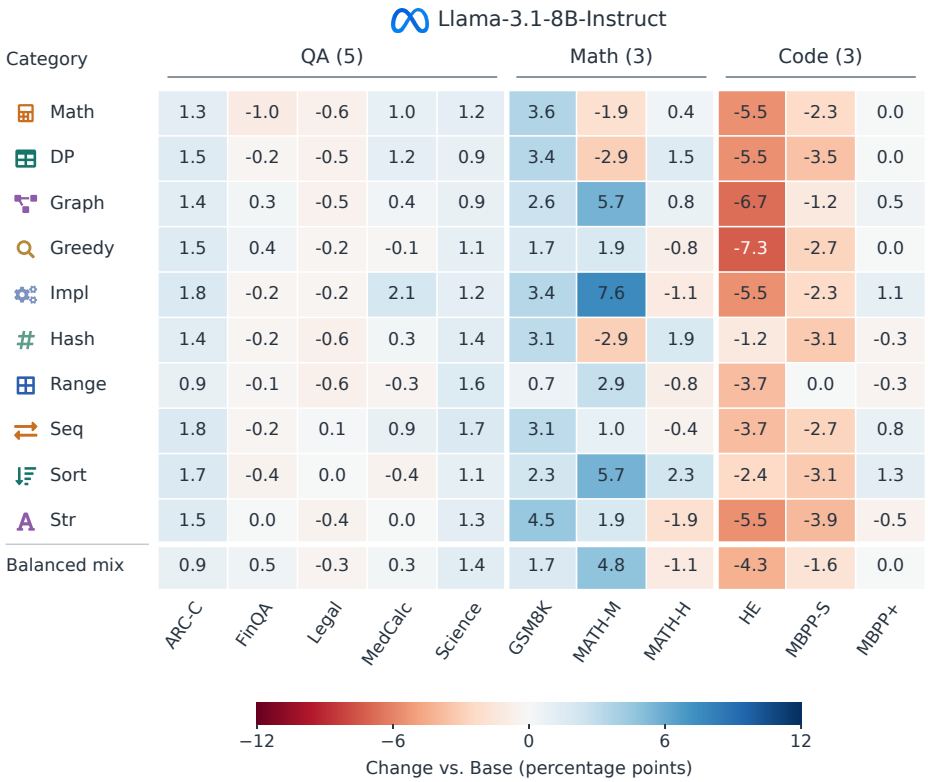


Figure 9: **Llama-3.1-8B-Instruct: complete task responses.**

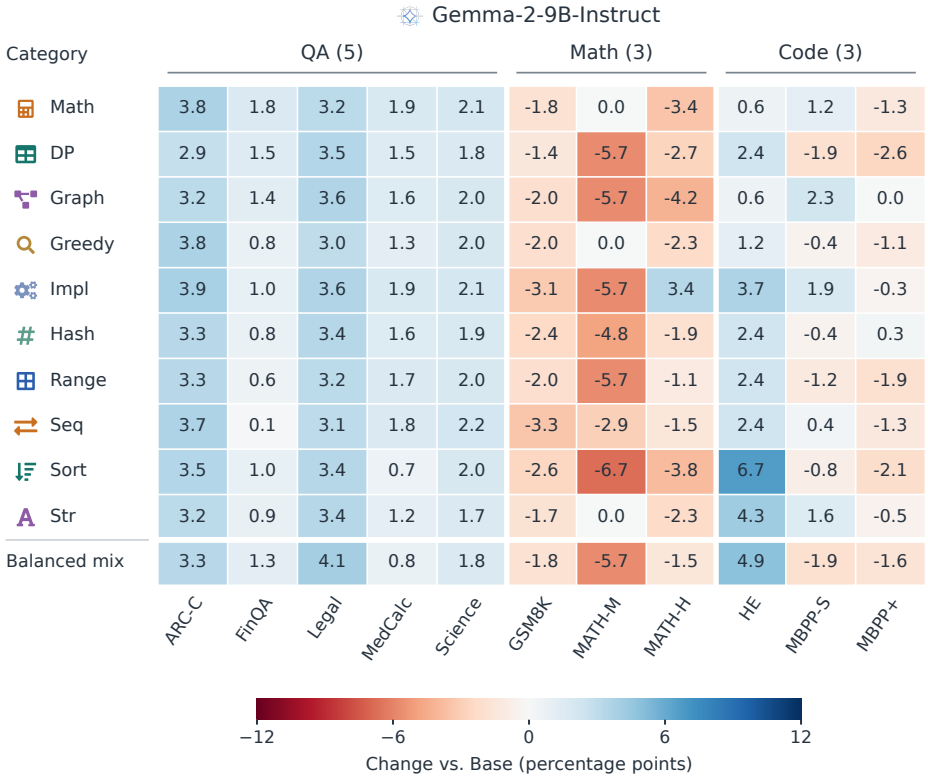


Figure 10: **Gemma-2-9B-Instruct: complete task responses.**

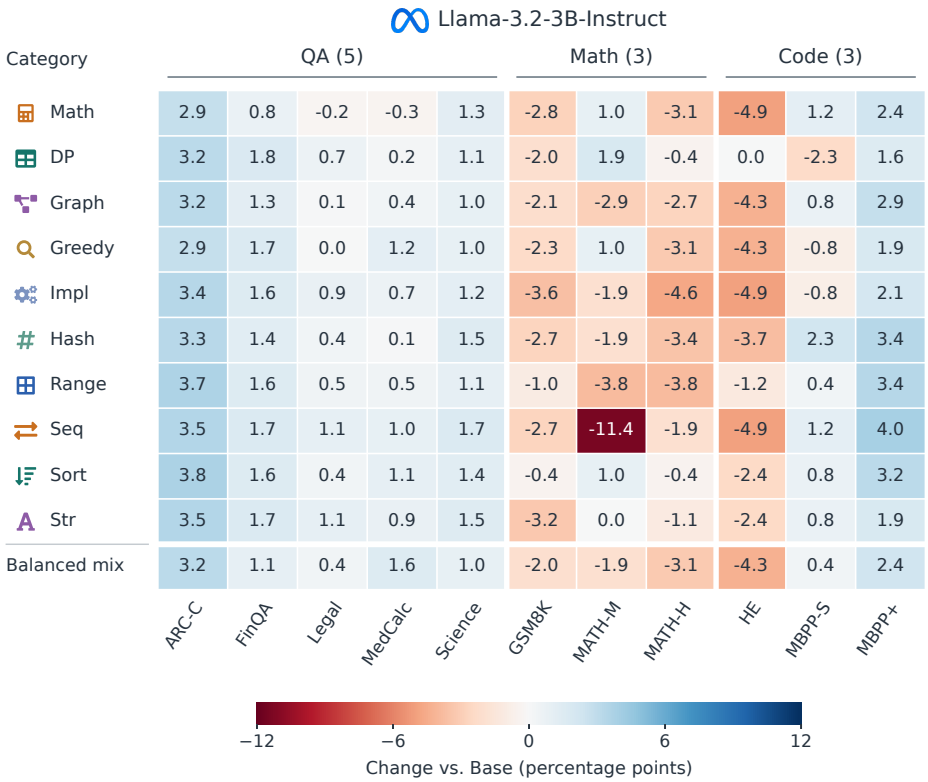


Figure 11: **Llama-3.2-3B-Instruct: complete task responses.**

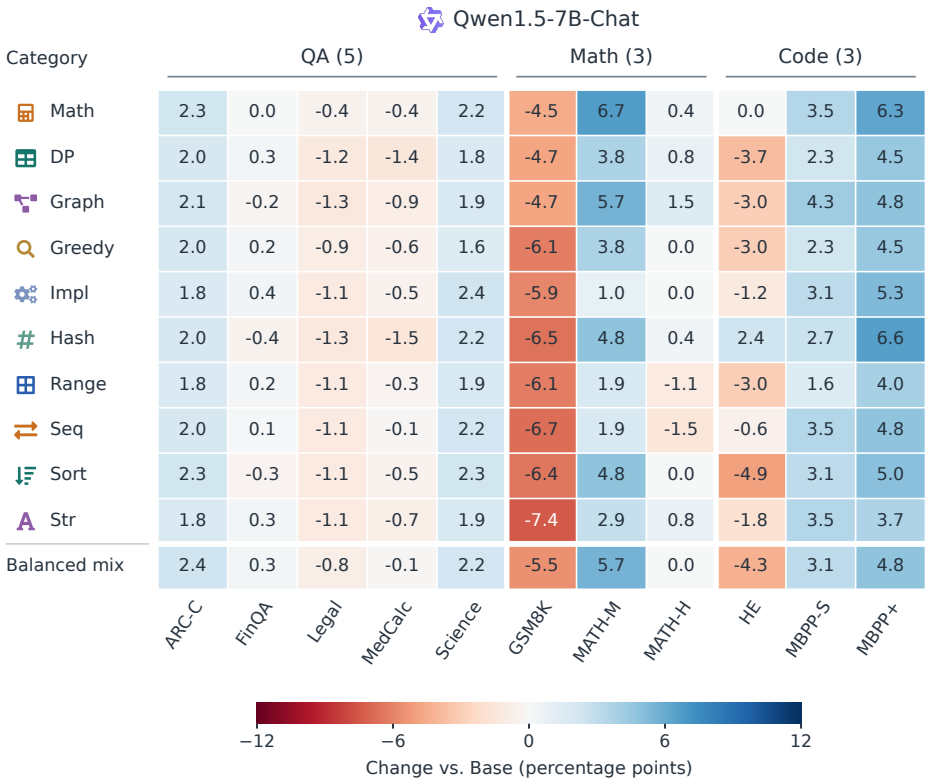


Figure 12: **Qwen1.5-7B-Chat: complete task responses.**

C Compact-Mixture Experiments and Further Validation

This experiment tests jointly trained mixtures suggested by the category responses. The initial experiment compares four compact recipes with a newly trained full-corpus reference on all six backbones. The comparisons are exploratory: model-specific categories were chosen using earlier results on the same eleven test configurations. The shared three-category recipe, shown in Figure 5b, was chosen heuristically from favorable single-category responses and complementary code patterns: direct implementation (Imp), # set/count manipulation (Hash), and ordered processing (Sort). Imp's QA-family gain is positive on all six backbones, and all three categories have high cross-backbone mean Overall scores in the preceding sweep. The recipe was fixed before mixture training; complementarity motivated testing the combination, without establishing additive effects or optimality.

C.1 Recipes, budgets, and reporting scope

Table 5 distinguishes the five trained arms from Base. The two Top3 recipes concatenate the earlier fixed 1,706-example category subsets and shuffle them. The balanced recipe samples from those category subsets; the proportional recipe samples within categories from the complete pool without replacement. All trained arms start from the original checkpoint; Complete is a fresh reference from this round, not an older full-training result or a continuation of a compact arm.

Table 5: **Mixture training arms.** Compact arms match examples and updates, but not token exposure. Complete uses the complete 35,974-example source pool.

Arm	Examples	Updates	Composition
Base	0	0	Original checkpoint
Personalized Top3	5,118	320	Three model-specific categories, 1,706 each
Shared Top3	5,118	320	Impl./util., # Hash/sets, Sort/order; 1,706 each
Balanced	5,118	320	Math and Seq: 511 each; other eight: 512 each
Proportional	5,118	320	Stratified sample preserving pool proportions
Complete	35,974	2,249	Full source pool, one epoch

Personalized Top3 takes the three highest earlier single-category Overall scores, not a QA-specific ranking. The selected sets are Imp/Str/# Hash (Qwen2.5), Seq/# Hash/Math (Qwen3), Math/# Hash/Graph (Qwen1.5), Sort/Imp/Graph (Llama-3.1), Sort/DP/Str (Llama-3.2), and Imp/Str/Math (Gemma-2). The proportional quotas are Imp 800, # Hash 722, Seq 632, Range 600, Str 567, DP 507, Graph 395, Sort 357, Math 295, and Greedy 243. Freezing these recipes before training does not make their evaluation independent of the earlier selection scores.

Training uses rank-16 LoRA with alpha 32, dropout 0.05, and q/k/v/o projections; learning rate 2×10^{-5} with cosine decay, warmup ratio 0.03, and weight decay 0.01; effective batch size 16; one epoch; BF16; maximum length 4,096; and assistant-only loss. The final adapter is evaluated with each model's native chat template and thinking disabled. Evaluation retains the original scoring rules, temperature 0.2 and top-p 0.95. Generation caps are 2,048 tokens for code, 256 for GSM8K, 512 for both MATH configurations, and 128 for FinQA/MedCalc. ARC, ScienceQA, and LegalBench use candidate likelihoods. The task denominators match Appendix A; the MATH subsets contain 105 and 262 questions.

The reported mixture means average seeds 20260730, 20260729, and 20270731, as confirmed by the authors; the third identifier intentionally differs from the earlier sweep. The accompanying report's detailed run paths and audit counts document seed 20260730. Individual scores for all seeds and raw predictions are absent from the supplied compact package, so these results do not quantify seed variability. The Overall tables preserve three decimals from the report, and task maps preserve one decimal from the source plots. The highlighted MATH contrasts use the report's finer-precision differences, not subtraction of rounded plot values. Machine-readable transcriptions, original primary score plots, and provenance notes accompany the source in `data/mixtures/`.

C.2 Overall scores and matched-budget comparisons

Table 6: **All mixture Overall means (%).** Overall equally weights eleven task percentages, with GSM8K at 256 tokens. Bold marks the largest trained-arm mean in each row; it does not denote statistical significance or necessarily an improvement over Base.

Model	Base	Personal.	Shared	Balanced	Proport.	Complete
Qwen2.5-7B	55.782	58.571	**59.165**	58.825	59.035	56.781
Qwen3-4B	55.242	56.872	56.361	57.361	56.651	**57.975**
Qwen1.5-7B	37.145	37.347	**38.407**	38.244	38.066	37.667
Llama-3.1-8B	51.496	52.077	51.726	52.363	51.589	**52.900**
Llama-3.2-3B	44.898	43.971	**44.264**	44.209	43.775	41.373
Gemma-2-9B	53.165	**53.988**	53.400	53.438	53.660	50.935

At least one compact recipe exceeds Complete on Overall for four of six backbones. Qwen3 and Llama-3.1 instead favor Complete; their best compact recipe is Balanced, trailing by 0.614 and 0.537 pp. Personalized exceeds Complete on three backbones, while Shared, Balanced, and Proportional each do so on four. Thus, an observed advantage over full-corpus training is not exclusive to category-based selection. The highest compact score is selected retrospectively from four candidates and is not an independently confirmed optimum.

Base remains essential to interpretation. All four Llama-3.2 compact arms exceed Complete yet fall below Base; Shared is 44.264 versus 44.898 for Base and 41.373 for Complete. Across all backbones, 20 of 24 compact arms raise Overall and all 24 raise the five-task QA mean. The QA-family direction does not imply gains on every constituent task. Figures 13 and 14 retain all task scores, including the decreases.

Table 7: **Top3 recipes versus equally sized controls (Overall, pp).** Each contrast is reported at the source precision. All arms here use 5,118 examples and 320 updates. Tiny differences are descriptive; no seed-stability claim is made.

Model	Personal.−Bal.	Personal.−Prop.	Shared−Bal.	Shared−Prop.
Qwen2.5-7B	-0.255	-0.464	$+0.340$	$+0.130$
Qwen3-4B	-0.489	$+0.221$	-1.000	-0.290
Qwen1.5-7B	-0.897	-0.720	$+0.164$	$+0.341$
Llama-3.1-8B	-0.286	$+0.488$	-0.636	$+0.137$
Llama-3.2-3B	-0.238	$+0.196$	$+0.055$	$+0.489$
Gemma-2-9B	$+0.550$	$+0.328$	-0.038	-0.260

Personalized Top3 exceeds Balanced only on Gemma-2 (one of six models); Shared does so on three. Both exceed Proportional on four. Even the favorable Gemma comparison combines a $+2.312$ pp Math-family change with a -0.659 pp Code-family change relative to Balanced. Ranking individual categories therefore does not guarantee that their combination retains every benefit. Differences from the earlier 1,706-example single-category training configurations also change the training budget and cannot establish category synergy or interference.

For Qwen2.5, Shared exceeds Complete by $+11.429$ pp on MATH Medium and $+9.542$ pp on MATH High. Displayed scores are 66.7 versus 55.2 and 34.7 versus 25.2; Base scores are 64.8 and 31.3. These are improvements over both references, but the gaps to Base are much smaller. Shared is also lower than Complete on HumanEval (-2.439 pp) and MBPP-Simple (-1.556 pp), and its Code-family mean is lower by about 0.979 pp. This is evidence for selected-task gains, not uniform replacement of full-corpus training.

C.3 GENERATION-BUDGET SENSITIVITY AND RESOURCE INTERPRETATION

GSM8K is additionally evaluated at a 1,024-token cap for Base and every trained arm on the same 1,319 questions. Table 8 shows that a larger budget especially benefits the Qwen2.5 and Qwen3 bases. Across the reported model–arm means, 34 of 36 absolute GSM8K scores increase, but trained-arm gains relative to the *same-budget* Base fall from 16 of 30 positive contrasts to 6 of 30 (Table 9). Ten positive contrasts become negative. These results remain separate from the primary eleven-task Overall; a single-task summary in the long-budget source plots is not that Overall.

Table 8: **GSM8K Base sensitivity to generation budget.** Positive-arm counts cover the five trained recipes per backbone.

Model	Base, 256	Base, 1,024	Change (pp)	Positive arms: 256→1,024
Qwen2.5-7B	76.346	91.054	+14.708	$5 \to 0$
Qwen3-4B	76.649	93.025	+16.376	$5 \to 1$
Qwen1.5-7B	59.591	63.154	+3.563	$1 \to 0$
Llama-3.1-8B	72.176	78.317	+6.141	$5 \to 5$
Llama-3.2-3B	72.328	76.801	+4.473	$0 \to 0$
Gemma-2-9B	84.685	85.444	+0.758	$0 \to 0$

Table 9: **GSM8K changes from same-budget Base (pp).** Each cell gives the 256-token change followed by the 1,024-token change. These are separately generated evaluations, not forced continuations.

Model	Personal.	Shared	Balanced	Proport.	Complete
Qwen2.5	$7.51 \to -1.36$	$9.93 \to -1.97$	$7.88 \to -1.21$	$8.57 \to -2.58$	$7.51 \to -2.12$
Qwen3	$10.46 \to -0.45$	$10.01 \to 0.30$	$9.33 \to -0.08$	$9.33 \to -0.38$	$13.12 \to -2.35$
Qwen1.5	$-2.12 \to -5.61$	$-2.81 \to -6.52$	$-2.81 \to -6.97$	$-2.43 \to -5.53$	$0.23 \to -2.50$
Llama-3.1	$3.49 \to 3.18$	$2.73 \to 0.99$	$3.34 \to 2.43$	$3.26 \to 2.27$	$5.99 \to 3.64$
Llama-3.2	$-2.73 \to -5.08$	$-5.31 \to -8.72$	$-6.29 \to -6.60$	$-3.64 \to -6.44$	$-23.12 \to -22.37$
Gemma-2	$-2.27 \to -1.67$	$-2.20 \to -1.90$	$-2.73 \to -2.05$	$-1.59 \to -1.29$	$-2.05 \to -1.74$

For Qwen2.5 Shared, the GSM8K advantage over Complete is $+2.426$ pp at 256 tokens but only about $+0.15$ pp at 1,024. All five Qwen2.5 arms fall below the long-budget Base; all five Llama-3.1 arms remain above it. The source audit also reports shorter mean responses for every trained arm at both caps, so brevity alone does not determine gain direction. The two budgets use separate sampled generations at temperature 0.2; identical seeds do not guarantee identical prefixes. Actual stop reasons and per-item tokenized-prompt hashes were not archived. The result supports budget sensitivity, without attributing every recovered answer to truncation or identifying a causal mechanism.

The compact arms use 14.23% of Complete's examples and updates. In the documented training audit, their final-training time ratios range from 13.77% to 16.26%, sequence-token ratios from 12.53% to 15.73%, and supervised-token ratios from 12.20% to 15.64%. These are recorded training-resource summaries, not three-seed uncertainty estimates or end-to-end savings. They exclude category screening, recipe search, and evaluation. Compact versus full comparisons also change token exposure and schedule length; the scores alone do not establish full-corpus overfitting or harmful categories.

C.4 FURTHER CONFIRMATION

Independent validation should fix a selection objective and category rule using separate selection data, then test the frozen recipes against equal-budget balanced and proportional samples on held-out tasks or questions. Paired seed records and token- or update-matched controls would help distinguish repeatability from training-budget effects. An equivalence or non-inferiority claim additionally requires a prespecified margin, uncertainty on the paired contrast, and full discovery-cost accounting. These stronger claims are left to future work.

C.5 Complete task scores

a Qwen2.5-7B

	QA				Mathematics				Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	51.5	68.0	18.4	72.3	6.2	76.3	64.8	31.3	81.7	75.1	68.0
Personalized	61.5	71.4	19.9	73.1	12.8	83.9	66.7	30.2	79.9	75.1	69.8
Shared	61.2	71.2	19.5	73.2	11.8	86.3	66.7	34.7	79.9	74.7	71.7
Balanced	61.3	71.3	19.8	72.8	13.1	84.2	64.8	32.1	79.9	77.4	70.4
Proportional	61.3	71.1	20.0	73.2	12.5	84.9	65.7	30.5	80.5	78.6	70.9
Complete	59.1	70.6	16.6	72.8	11.9	83.9	55.2	25.2	82.3	76.3	70.6

b Qwen3-4B

	QA				Mathematics				Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	45.2	59.4	15.9	83.3	11.6	76.6	48.6	23.3	86.6	83.3	73.8
Personalized	54.4	68.0	15.0	85.3	13.5	87.1	53.3	24.8	78.0	76.3	69.8
Shared	54.4	67.8	15.5	85.5	12.6	86.7	50.5	22.9	76.8	76.7	70.6
Balanced	53.5	67.8	16.2	85.2	13.0	86.0	55.2	24.4	81.7	77.8	70.1
Proportional	54.1	67.9	15.9	85.1	14.1	86.0	51.4	25.2	79.3	75.1	69.0
Complete	57.2	69.1	16.1	84.3	14.3	89.8	53.3	27.1	79.3	77.8	69.6

c Qwen1.5-7B

	QA				Mathematics				Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	54.9	67.5	7.0	47.2	4.2	59.6	21.0	6.9	45.7	51.4	43.4
Personalized	57.0	69.7	7.2	46.9	3.6	57.5	20.0	8.8	40.9	52.9	46.3
Shared	57.3	69.8	6.4	45.8	3.6	56.8	26.7	7.6	44.5	56.0	47.9
Balanced	56.7	69.4	7.1	46.7	4.5	56.8	23.8	7.6	47.6	52.9	47.6
Proportional	56.7	69.2	7.1	46.4	3.6	57.2	26.7	6.9	47.0	51.0	47.1
Complete	54.7	66.4	8.5	46.7	3.5	59.8	25.7	7.6	41.5	52.9	47.1

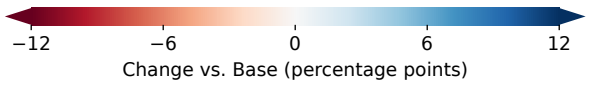


Figure 13: **Complete mixture task scores: Qwen backbones.** All six original arms and eleven task configurations are shown as reported three-seed mean percentages, at the source plots' one-decimal precision. Base rows are unshaded; other cells are blue/red for gains/declines relative to the same model's Base. The shared red–blue scale matches Figure 12 and saturates beyond ±12 pp, as indicated by the colorbar extensions. Shading uses differences between the displayed one-decimal scores; cell labels remain absolute percentages. Shared combines Impl./util., # Hash/sets, and Sort/order. The primary GSM8K cap is 256 tokens; MATH M/H denote the Medium/High configurations. Overall is reported separately in Table 6.

d Llama-3.1-8B

	QA					Mathematics			Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	50.3	67.0	11.7	81.4	13.7	72.2	45.7	23.3	67.7	70.8	62.7
Personalized	51.6	67.9	12.4	80.9	14.1	75.7	47.6	24.4	64.6	70.0	63.5
Shared	52.0	68.0	10.8	80.8	14.3	74.9	44.8	25.2	65.2	69.6	63.5
Balanced	52.0	68.5	11.1	80.8	15.0	75.5	53.3	23.3	66.5	66.9	63.0
Proportional	51.7	68.0	11.4	80.8	15.1	75.4	46.7	23.3	63.4	68.5	63.2
Complete	49.6	65.8	14.1	79.3	14.5	78.2	55.2	26.0	62.2	71.6	65.3

e Llama-3.2-3B

	QA					Mathematics			Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	43.6	65.7	8.9	56.1	6.9	72.3	44.8	23.3	54.3	61.9	56.1
Personalized	46.6	67.0	9.8	56.4	7.7	69.6	37.1	19.5	52.4	59.1	58.5
Shared	46.6	66.6	9.3	55.9	7.2	67.0	41.0	21.8	54.3	59.1	58.2
Balanced	46.6	66.0	9.3	55.4	7.5	66.0	43.8	22.9	52.4	58.8	57.7
Proportional	46.7	66.1	9.4	56.0	7.0	68.7	40.0	23.3	50.0	57.6	56.9
Complete	40.5	60.7	14.0	55.1	7.3	49.2	38.1	23.7	47.6	62.3	56.6

f Gemma-2-9B

	QA					Mathematics			Code		
	ARC	Science QA	FinQA	Legal Bench	MedCalc	GSM8K	MATH M	MATH H	Human Eval	MBPP Simple	MBPP+
Base	56.7	70.4	19.5	62.6	13.1	84.7	60.0	28.6	59.1	66.5	63.5
Personalized	61.3	72.3	21.6	64.7	15.0	82.4	58.1	27.9	62.8	66.1	61.6
Shared	60.2	72.0	20.5	65.4	15.3	82.5	53.3	27.1	62.2	66.5	62.4
Balanced	60.5	72.3	21.3	65.2	14.5	82.0	54.3	25.2	62.8	66.5	63.2
Proportional	60.8	72.3	21.0	65.4	14.7	83.1	56.2	26.3	61.0	68.1	61.4
Complete	57.0	69.6	18.4	64.2	15.0	82.6	50.5	28.2	58.5	57.2	59.0

−12 −6 0 6 12

Change vs. Base (percentage points)

Figure 14: **Complete mixture task scores: Llama and Gemma backbones.** Presentation, precision, and the ±12 pp color scale follow Figure 13. Base rows remain unshaded. Base and Complete are retained alongside every compact recipe so that reduced degradation can be distinguished from improvement over the starting checkpoint. No seed-level error bars or significance tests are inferred from these means.

C.6 TASK-LEVEL SCREENING AGAINST CONSTITUENT CATEGORIES AND COMPLETE

Let $\bar{S}^{a}_{b,t}$ denote the three-seed mean percentage score of arm a for backbone b on task t. For backbone b, task t, and a set of k categories C ($k = 2$ or 3), the broader screen requires

$$\bar{S}^{\mathrm{mix}(C)}_{b,t} > \max_{c \in C} \bar{S}^{c}_{b,t}, \qquad \bar{S}^{\mathrm{mix}(C)}_{b,t} > \bar{S}^{\mathrm{Complete}}_{b,t}. \tag{2}$$

Here $\mathrm{mix}(C)$ is one jointly trained model; each constituent is trained separately. Complete uses 35,974 examples. To additionally require every constituent to improve on Base, we impose

$$\bar{S}^{\mathrm{Base}}_{b,t} < \min_{c \in C} \bar{S}^{c}_{b,t}. \tag{3}$$

The tables retain all broader-screen passes and bold mixture scores that also meet Equation 3. Scores are reported three-seed means, as confirmed by the authors. Top2 uses seeds 20260730, 20260729, and 20260731; the earlier Top3 round uses 20260730, 20260729, and 20270731. Broader decisions use the reports' full-precision comparisons; tables show two decimals. Per-seed arrays were not supplied for independent reconstruction or uncertainty estimation.

For three categories, the broader screen retains $21/132$ comparisons: ten Personalized and eleven Shared results across 17 model–task pairs; nineteen exceed Base. The additional condition retains $18/132$ across 14 model–task pairs. Excluding GSM8K gives $20/120$ broader passes and $17/120$ stricter passes.

Table 10: **All 21 broader three-category passes.** Best single is the highest constituent score. Scores are percentages; ΔBase is in pp. Bold mixture scores mark the 18 passes for which every constituent also exceeds Base.

Backbone	Task	Recipe	Best single	Mixture	Complete	ΔBase
Qwen2.5	ARC-C	Imp + Str + Hash	59.64	**61.52**	59.13	+10.07
	ARC-C	Imp + Hash + Sort	59.64	**61.18**	59.13	+9.73
	ScienceQA	Imp + Str + Hash	70.59	**71.45**	70.64	+3.42
	ScienceQA	Imp + Hash + Sort	70.59	**71.18**	70.64	+3.15
	GSM8K†	Imp + Hash + Sort	84.08	**86.28**	83.85	+9.93
	MATH-H	Imp + Hash + Sort	30.15	34.73	25.19	+3.44
	FinQA	Imp + Str + Hash	19.33	**19.95**	16.59	+1.50
	FinQA	Imp + Hash + Sort	19.33	**19.51**	16.59	+1.06
	MedCalc	Imp + Str + Hash	11.00	**12.82**	11.91	+6.64
Qwen3	LegalBench	Seq + Hash + Math	84.67	**85.26**	84.25	+1.95
	LegalBench	Imp + Hash + Sort	84.67	**85.49**	84.25	+2.19
Qwen1.5	ARC-C	Imp + Hash + Sort	57.17	**57.34**	54.69	+2.47
	ScienceQA	Math + Hash + Graph	69.69	**69.74**	66.37	+2.25
	LegalBench	Math + Hash + Graph	46.83	46.89	46.71	−0.30
	MATH-M	Imp + Hash + Sort	25.71	**26.67**	25.71	+5.71
	MATH-H	Math + Hash + Graph	8.40	**8.78**	7.63	+1.91
	MBPP-Simple	Imp + Hash + Sort	54.47	**56.03**	52.92	+4.67
Llama-3.2	HumanEval	Imp + Hash + Sort	51.83	54.27	47.56	0.00
Gemma-2	ARC-C	Imp + Str + Math	60.67	**61.26**	57.00	+4.52
	FinQA	Imp + Str + Math	21.27	**21.62**	18.36	+2.12
	MedCalc	Imp + Hash + Sort	15.00	**15.27**	15.00	+2.18

† GSM8K uses the 256-token primary protocol. The source lacks the full set of constituent single-category scores at 1,024 tokens, so this pass is not established under the larger budget. All other tasks retain their original evaluation settings.

The stricter condition excludes Qwen2.5 Shared on MATH-H, Qwen1.5 Personalized on LegalBench, and Llama-3.2 Shared on HumanEval. In the latter two cases, the mixture itself is below or tied with Base. On Qwen2.5 MATH-H, the mixture improves on Base even though its constituents individually decline, illustrating a pattern excluded by the additional condition.

Table 11: **All 11 broader two-category passes.** Top2 retains the two leading categories in each backbone's earlier Overall ranking. Bold mixture scores mark the nine passes for which both constituents exceed Base. Other columns follow Table 10.

Backbone	Task	Recipe	Best single	Mixture	Complete	ΔBase
Qwen2.5	ARC-C	Imp + Str	59.64	**60.75**	59.13	+9.30
	ScienceQA	Imp + Str	70.59	**70.77**	70.64	+2.74
	GSM8K†	Imp + Str	84.00	**84.84**	83.85	+8.49
Qwen3	LegalBench	Seq + Hash	84.67	**85.08**	84.25	+1.78
	MBPP-Simple	Seq + Hash	78.60	79.38	77.82	−3.89
Qwen1.5	ARC-C	Math + Hash	57.17	**57.76**	54.69	+2.90
	ScienceQA	Math + Hash	69.69	**69.96**	66.37	+2.47
	MATH-H	Math + Hash	7.25	**8.40**	7.63	+1.53
Llama-3.1	ARC-C	Sort + Imp	52.05	**52.30**	49.57	+2.05
	ScienceQA	Sort + Imp	68.17	**68.21**	65.83	+1.26
Llama-3.2	HumanEval	Sort + DP	54.27	55.49	47.56	+1.22

† The same GSM8K budget qualification applies. The Qwen2.5 Top2 score is below the same-budget Base at 1,024 tokens; the screening conditions cannot be checked there without the constituent scores.

For two categories, the broader screen retains $11/66$ comparisons across five backbones, ten of which exceed Base. The additional condition retains $9/66$, excluding Qwen3 on MBPP-Simple and Llama-3.2 on HumanEval. Excluding GSM8K gives $10/60$ broader passes and $8/60$ stricter passes. These fractions have different candidate sets and constituent thresholds from the three-category screen; they do not compare interaction strengths across mixture sizes. Across both mixture sizes, the 27 stricter passes cover 16 distinct model–task pairs, five backbones, and nine evaluation configurations.

The additional condition is checked against the released one-decimal Base-relative task changes. A displayed 0.0 or -0.0 does not establish an exact tie. The only such value here is Llama-3.2's DP result on HumanEval, whose Sort constituent already declines; therefore no retained-count decision depends on an unresolved zero. All retained constituents have strictly positive displayed gains.

Two Qwen1.5 examples satisfy all conditions. On ARC-C, Math (57.17%) and Hash (56.83%) each exceed Base (54.86%); their Top2 mixture reaches 57.76%, above both constituents and Complete (54.69%). On MBPP-Simple, Imp, Hash, and Sort score 54.47%, 54.09%, and 54.47%, each above Base (51.36%); Shared reaches 56.03%, versus Complete's 52.92%. The margins over the best constituent are 0.60 and 1.56 pp. Both mixtures also exceed the existing Balanced and Proportional controls, although those 5,118-example controls are larger than Top2.

All selections are exploratory. Category rankings used the same test suite, and the task-level screen was applied after observing results. Single-category training uses 1,706 examples and 107 updates; Top2 uses 3,412 and 214; Top3 uses 5,118 and 320; Complete uses 35,974 and 2,249. Token exposure is also unmatched. Consequently, exceeding constituent models does not isolate category synergy from training amount, and exceeding Complete does not establish optimality or statistical significance. Independent recipe selection and matched-budget confirmation remain future work.

The accompanying CSVs preserve the broader passes, constituent Base-relative changes, and stricter subsets. A supplied Python script reproduces this audit and flags unresolved rounding boundaries without reconstructing per-seed results.

D EXPLORATORY AUDIT OF ⚙ DIRECT IMPLEMENTATION AND QA TRANSFER

This appendix examines a possible interpretation of the ⚙ Imp response in RQ2: ⚙ direct implementation and utilities (Imp) may help a model apply stated conditions and variable relationships more faithfully. The analysis describes a secondary transfer phenomenon and its possible contributors. It does not identify a causal mechanism.

D.1 AUDIT SCOPE AND COMPARATIVE GAINS

The audit covers one documented training seed, 20260730, with six backbones and twelve checkpoints per backbone: Base, ten single-category training configurations, and the balanced mixture. It uses 1,172 ARC-Challenge and 2,224 ScienceQA questions, yielding $72 \times (1{,}172 + 2{,}224) = 244{,}512$ saved choice predictions. Their selected options and correctness agree with the saved candidate scores and gold answers under the original scoring rule. These records supplement the reported three-seed summaries in the main analysis; they do not supply the missing seed-level variation. All averages below are descriptive means across the six fixed backbones.

Table 12 separates improvement over Base from advantage over other code categories. ⚙ Imp improves both benchmarks on every backbone in this audit. Its mean advantage over the other nine single-category training configurations is smaller and has exceptions. The other-nine comparator is the arithmetic mean of nine independently trained arms, not a voting ensemble; the balanced mixture is excluded from that mean.

Table 12: **Single-seed ⚙ Imp contrasts.** Accuracy differences in pp for seed 20260730. The final row averages the six backbones; it is not a training-replication estimate.

	ARC-Challenge		ScienceQA	
Backbone	⚙ Imp−Base	⚙ Imp−Other nine	⚙ Imp−Base	⚙ Imp−Other nine
Qwen2.5-7B	+8.19	+2.28	+2.56	+0.33
Qwen3-4B	+6.23	+0.84	+5.53	+1.51
Llama-3.1-8B	+1.79	+0.36	+1.21	−0.04
Gemma-2-9B	+3.92	+0.52	+2.07	+0.11
Llama-3.2-3B	+3.41	+0.08	+1.21	−0.08
Qwen1.5-7B	+1.79	−0.24	+2.38	+0.39
Mean	+4.22	+0.64	+2.50	+0.37

D.2 ILLUSTRATIVE CASES AND THE SCORING PROTOCOL

⚙ Direct-implementation training includes tasks that turn a natural-language specification into variable bindings, threshold checks, and aggregation. For example, training item `Algorithm_10021_C` maps shipping type and quantity to a price, changes the rate at a ten-item threshold, and sums the costs. Such operations motivate the hypothesis that training can help apply existing knowledge to a question's conditions. Table 13 gives selected answer changes compatible with this interpretation, together with a factual improvement and a relational regression. Cases were selected after inspecting outcomes; their frequency cannot be inferred from this table.

ARC-Challenge and ScienceQA score each complete candidate answer, including its leading space, by mean token log probability and select the highest-scoring option. ScienceQA uses text and available hints without images. The template is fixed across arms within each backbone, and no explanation is generated. Consequently, these gains cannot arise directly from shorter generated explanations avoiding truncation or answer-extraction failures. This check does not exclude changes in option-length preference, wording preference, or likelihood calibration.

The reasoning descriptions in Table 13 are analyst interpretations of the question and selected answer, not decoded reasoning traces. Some changes occur near a decision boundary: the average-speed example has an ⚙ Imp top-two score gap of only 0.0116 mean log-probability units. Other inspected

positives involving salinity and net force have gaps of 0.00375 and 0.0128. A tied-score case is also present in the audit, so unique correctness under the recorded tie-breaking rule alone is not strong evidence of acquired knowledge.

Table 13: **Selected improvements and counterexamples.** Answers are paraphrased from saved predictions. R/F descriptions indicate the analyst's interpretation. These illustrative cases do not establish a mechanism or a prevalence estimate.

Backbone / item	Question and Base→Imp answer	Interpretation and comparison
Llama-3.1 / ARC `arc_00372`	One hour at 80 km/h, then one hour at 100 km/h: 80 → 90 km/h.	Corrected equal-duration average speed; all other nine arms and the balanced mixture are wrong.
Qwen2.5 / ARC `arc_00564`	A substance expands and then melts when heated: initial state liquid→solid.	Corrected initial-state inference; all other nine arms and the balanced mixture are wrong.
Llama-3.2 / ScienceQA `scienceqa_02187`	Given dominant E, recessive e, and genotype ee: brown→red eyes.	Corrected rule application; all other nine arms and the balanced mixture are wrong.
Qwen2.5 / ARC `arc_00821`	Imp correctly selects that plants existed before coal and oil formed; Base is wrong.	Factual improvement; all other nine single-category training configurations are wrong.
Llama-3.1 / ScienceQA `scienceqa_02144`	Both journeys take five hours: 75 miles→60 miles as the faster journey.	Relational regression: Base is correct and Imp is wrong.
Qwen2.5 / ScienceQA `scienceqa_01144`	Identify an object that is not a mineral: skull→quartz.	Classification regression: Base and all other nine arms are correct.

D.3 Do additional gains concentrate in relational questions?

We inspect this prediction with a fixed sample of 80 questions per benchmark. Questions are sorted by the SHA256 hash of `imp-blind-20260924|task|question-id`, and the first 80 are retained, without selecting by Imp correctness. A single AI annotator labels the question text as explicit relation/rule application (R), factual/definitional knowledge (F), language-specific rules (L), or other/ambiguous (O). Labels and reasons are retained, but there is no independent human adjudication. This is an exploratory, post hoc analysis, not a preregistered or independently verified double-blind study.

For question i, let $y_{bci} \in \{0, 1\}$ be correctness for backbone b and arm c. We first average each question's Imp advantage over the six fixed backbones:

$$d_i = \frac{100}{6} \sum_{b=1}^{6} \left(y_{b,\text{Imp},i} - \frac{1}{9} \sum_{c \neq \text{Imp}} y_{bci} \right),$$

where the inner sum includes only the other nine single-category training configurations. We then compare $\bar{d}_R - \bar{d}_F$, resampling questions within each group for 20,000 bootstrap replicates (resampling seed 20260924). The percentile intervals quantify question-sampling uncertainty conditional on these models and labels. They do not include training randomness, annotation error, or uncertainty from choosing the hypothesis after observing the results.

Table 14: **Exploratory question-type comparison.** R/F gains are Imp minus the other-nine mean, in pp. The intervals apply to R−F, not to training-seed variability.

Benchmark	R/F n	R gain	F gain	R−F	95% interval
ARC-Challenge	26/41	+0.28	+2.89	−2.61	[−4.76, −0.54]
ScienceQA	20/15	+0.65	+1.48	−0.83	[−3.95, +2.04]

This sample does not support the prediction that Imp's additional gains concentrate in relational questions: the ARC-Challenge difference is in the opposite direction, and the ScienceQA interval

spans zero. The groups are small, the R definition excludes language-specific rules, and questions are not matched for difficulty or Base accuracy. The result therefore constrains the proposed explanation without establishing the absence of any benefit to rule application.

ScienceQA's full evaluated subset contains 1,043 natural-science, 1,056 language-science, and 125 social-science questions. Its total score should not be equated with natural-science reasoning. Grouping all natural-science questions by their original skill metadata provides a complementary check (Table 15). Physical-relation and experimental-constraint questions show small advantages over the other categories while declining relative to Base. Thus, relative category preference can coexist with reduced absolute performance on a proposed mechanism-relevant subset.

Table 15: **ScienceQA natural-science skill groups.** Descriptive six-backbone mean changes in pp, using all questions in each indicated metadata group. "Remaining" questions are not a pure factual control.

Skill group	n	Imp−Base	Imp−Other nine
Physical relations (e.g., speed, force, energy)	141	−0.35	+0.64
Experimental constraints	61	−1.37	+0.82
Genotype, phenotype, and dominance rules	105	+2.38	+0.67
Inherited/acquired trait evidence	113	+1.33	+0.02
Remaining natural-science questions	623	+4.07	+0.52

D.4 Training content, response style, and open questions

The 1,706 Imp examples include aggregation (509), simple conversion (401), validation (368), mapping/filtering (167), library/API use (138), formatting (69), file-system operations (30), and date/time operations (24). These labels do not isolate a cognitive operation. Some formatting examples apply department and tenure thresholds, while some prompts already provide example implementations. Moreover, Python AST conditionals occur in 969/1,706 Imp examples (56.8%), compared with 1,605/1,706 dynamic-programming examples (94.1%) and 1,572/1,706 graph examples (92.1%). The records therefore do not support the simple explanation that Imp contains a larger share of examples with conditionals; structural counts do not measure the quality of rule-application supervision.

The arms match 1,706 examples and 107 updates, but not token exposure. For Qwen2.5, Imp uses 643,839 total input tokens and 208,684 supervised tokens, compared with 772,561/284,522 for dynamic programming and 980,483/326,472 for graph training. These are overlapping total and supervised counts, not quantities to add. Content, length, and supervision density remain coupled in the current comparison.

Generated-response records also show task-dependent style changes. For Qwen3, mean visible GSM8K length decreases from 181.5 to 163.7 tokens, while FinQA increases from 40.2 to 43.8 and MedCalc from 39.4 to 51.7. In one complete GSM8K response, Imp applies a stated attendance fraction once, correcting Base's repeated application; in another, both responses derive the same four quantities but Imp corrects their sum. These selected examples compare only Base with Imp and cannot establish an Imp-specific advantage over other categories. Visible lengths are reconstructed from saved text and do not recover original decoding stop reasons.

Taken together, the audit documents consistent Base-relative gains on these two QA benchmarks and examples compatible with improved application of stated conditions. It does not establish relational selectivity or a shared mechanism across backbones. Isolating that explanation would require training subsets annotated for the proposed operations, comparisons with matched token exposure and update counts, repeated training, and evaluation held out from hypothesis and data selection. We leave those controlled causal tests to future work.

The accompanying `data/imp_qa_audit/` directory preserves the summary tables, question sample and labels, training-budget and structure summaries, and saved case records used in this appendix. Its README distinguishes these derived records from the full prediction archive and from the main three-seed score summaries.

E AUDIT OF QA COVERAGE AND BACKBONE-DEPENDENT RESPONSES

This appendix examines the scope of RQ1 and possible contributors to its task-dependent responses. It uses saved outputs to distinguish task composition, changes in solutions, and evaluation-interface effects. The main results retain the reported three-seed aggregates; this complementary audit covers seed 20260730 only and does not independently verify the other two runs or estimate training variability.

E.1 RECORDS AND PROTOCOL CONSISTENCY

The audited run contains six Base checkpoints, sixty single-category adapters, and six balanced-mixture adapters. All 66 training records document 1,706 examples, 107 optimizer steps, and the same LoRA settings. Evaluation contracts, templates, scoring populations, and decoding parameters match across the twelve arms within each backbone. Category token exposure is not matched. These checks support within-run comparisons; they do not isolate training-content effects from all other sources of variation.

QA counts and denominators agree across the saved metrics and 360 task-output files. The three mathematics configurations contain 121,392 saved outputs and 111,276 Base–trained question pairs; the three code configurations contain 57,528 outputs and 52,734 pairs. Paired outcomes are denoted CC (both correct), CW (Base correct, trained wrong), WC (Base wrong, trained correct), and WW (both wrong), according to the original scorer. The same Base response and question recur across category comparisons, so summed transition counts are checkpoint–question comparisons, not independent observations. Descriptive category means below exclude the balanced mixture, whose records are retained separately.

E.2 HOW BROADLY DO THE QA GAINS EXTEND?

Table 16 recomputes task-family changes from full-precision correct-count records. All sixty single-category comparisons improve the five-task mean, but only 35 improve every constituent task individually. Removing ARC-Challenge and ScienceQA leaves 45 positive comparisons. Llama-3.1's remaining-three mean is close to zero, and Qwen1.5's is negative, whereas the other four backbones retain positive means in all ten categories.

Table 16: **QA composition sensitivity in seed 20260730.** Positive-arm counts and descriptive changes in pp. QA5 weights ARC-Challenge, FinQA, LegalBench, MedCalc, and ScienceQA equally; QA3 retains FinQA, LegalBench, and MedCalc. Changes average ten single-category training configurations per backbone.

	Positive arms		Mean change (pp)	
Backbone	QA5	QA3	QA5	QA3
Qwen2.5-7B	10/10	10/10	+2.79	+1.85
Qwen3-4B	10/10	10/10	+2.39	+0.77
Llama-3.1-8B	10/10	5/10	+0.55	+0.004
Gemma-2-9B	10/10	10/10	+2.26	+1.95
Llama-3.2-3B	10/10	10/10	+1.45	+0.86
Qwen1.5-7B	10/10	0/10	+0.47	−0.57
All comparisons	60/60	45/60	+1.65	+0.81

Removing ARC-Challenge alone leaves 57/60 positive comparisons; removing ScienceQA alone leaves 56/60. Removing FinQA, LegalBench, or MedCalc individually leaves 60/60 in each case. Thus, the universal direction of the five-task mean partly depends on task composition, although substantial positive transfer remains beyond ARC-Challenge and ScienceQA. These counts are not rounding artifacts. They describe the specified benchmark configurations and do not establish uniform gains across QA domains or a particular reasoning mechanism. ScienceQA's evaluated subset also includes language- and social-science questions (Appendix D).

E.3 GSM8K: SHORTER RESPONSES ACCOMPANY OPPOSITE SCORE CHANGES

The single-seed GSM8K directions match RQ1: every category improves Qwen2.5, Qwen3, and Llama-3.1, while every category lowers Gemma-2, Llama-3.2, and Qwen1.5. Nevertheless, all sixty arms have shorter mean visible responses than their corresponding Base (Table 17). Output shortening alone therefore does not distinguish the positive and negative responses.

Table 17: **GSM8K score and length changes.** Seed 20260730; $n = 1{,}319$ questions per arm. Trained scores and lengths average ten category arms, not repeated training runs. Visible tokens are obtained by retokenizing saved text.

Backbone	Mean Δ (pp)	Positive/negative	Base tokens	Trained tokens
Qwen2.5-7B	$+7.09$	10/0	193.3	148.7
Qwen3-4B	$+3.71$	10/0	181.5	168.6
Llama-3.1-8B	$+2.84$	10/0	177.1	170.6
Gemma-2-9B	-2.24	0/10	100.0	89.0
Llama-3.2-3B	-2.27	0/10	175.6	160.9
Qwen1.5-7B	-5.91	0/10	169.4	113.7

Replaying the archived GSM8K numeric scoring rule reproduces all 94,968 saved correctness flags. A separate sensitivity analysis takes the last explicit answer marker among `answer is`, `###`, and boxed answers, falling back to the original rule if none exists. Applied uniformly to Base and every trained arm, it preserves all sixty single-category gain directions. However, it changes 440 originally correct records to incorrect and restores none. Some outputs literally end with the placeholder `The answer is N.`, for which the original rule can recover a numeric answer elsewhere in the response. The alternative is therefore a sensitivity check, not a validated correction or evidence that extraction errors are absent.

The case review identifies both kinds of scoring mismatch. Qwen2.5 Base correctly calculates $8.75 - (-6.25) = 15$ on `gsm8k_00589`, but numbers in an unfinished closing summary interfere with extraction. Conversely, Qwen3 Base does not finish the requested calculation on `gsm8k_00150`, yet a step number is extracted as the reference answer, 4. Original scores are retained for both cases. MATH-M and MATH-H use the original symbolic/numeric scoring outcomes; this audit does not independently implement a replacement symbolic scorer.

E.4 CODE FAILURES INCLUDE SEMANTIC AND INTERFACE CHANGES

On HumanEval, Qwen3 has 154 CW and 36 WC comparisons across its ten categories, whereas Gemma-2 has 85 CW and 129 WC comparisons. Table 18 shows that aggregate score changes can conceal different failure profiles. An assertion failure records a failed test, not a diagnosis that every such failure has the same semantic cause.

Table 18: **HumanEval transitions and recorded failure types.** Seed 20260730; sums over ten categories, each evaluated on 164 questions. The last four columns partition CW comparisons. Prompt-helper and indentation entries are compatibility diagnoses, not verified recoverable scores.

Backbone	CW	WC	Assertion	Helper	Indentation	Other
Qwen3-4B	154	36	131	10	0	13
Gemma-2-9B	85	129	53	5	27	0

For HumanEval/50, a helper supplied in the prompt is sometimes omitted when the evaluator extracts a complete target function. Tests then fail because the helper is unavailable. Other Gemma-2 outputs provide a function body whose outer indentation is not restored correctly by the wrapper. Static checks show that adding the missing outer indentation can make such code compilable, but the repaired programs were not executed in this audit. A syntax repair therefore cannot be counted as a recovered correct answer. All original test outcomes remain unchanged.

Shortening also occurs in both directions within the same code benchmark: 150/154 Qwen3 HumanEval regressions (97.4%) and 33/36 improvements (91.7%) have shorter visible outputs than Base. Nor does every backbone shorten code responses: Gemma-2's mean HumanEval visible length rises from 67.5 to 87.1 tokens while its extracted-code character count falls, partly because extraction can include prompt material. These are different measurements, and neither alone establishes why correctness changes.

E.5 Illustrative cases and interpretation limits

The audit uses fixed samples of 24 GSM8K pairs (one per backbone and CC/CW/WC/WW stratum) and 48 HumanEval pairs (two per stratum). Model/category names are concealed and A/B order is permuted before a single AI review; labels are frozen before unblinding. This is not independent human adjudication or a representative estimate of error-type prevalence. Table 19 selects examples from those records to illustrate different sources of change.

Table 19: **Examples of solution and interface changes.** Seed 20260730; analyst interpretations of saved outputs. Category names identify the inspected arm and do not imply that the change is unique to that category.

Backbone / arm / item	Observed change and interpretation
Qwen1.5 / A String `gsm8k_00972`	Base confuses 40 oranges kept with the number sold; the trained response correctly gives 120 sold. Both answers are complete, and the correct response is shorter (117→105 visible tokens).
Gemma-2 / Sorting `gsm8k_01070`	The trained response halves the quantity of discounted butter twice, changing the correct answer 18 to 15. The response becomes longer (87→112 tokens).
Qwen3 / Imp HumanEval/26	Base removes every value occurring more than once; the trained code instead retains one copy of each value. This changes required semantics. The same regression occurs in all ten categories and the mixture on this item.
Gemma-2 / Sequence HumanEval/159	A shorter program correctly updates remaining inventory by subtracting the available amount sold. This is a successful relation update rather than a length-induced failure.
Gemma-2 / Sorting HumanEval/50	The decoding formula is retained, but the prompt-provided helper is absent from the assembled test program. The recorded regression is compatible with a wrapper failure.

These observations motivate a hypothesis that code training changes how existing solution strategies apply constraints and produce outputs, with consequences that depend on the backbone and evaluation interface. Individual cases exhibit actual relation or program-logic changes, while other outcomes depend on extraction or packaging. The audit does not identify their population-level causal contributions, and brevity alone cannot account for both improvements and regressions.

Original generation token IDs and stopping reasons are unavailable. Retokenized lengths near the generation cap are diagnostic proxies, not confirmed truncations. Subsets defined by both models producing short or explicit answers condition on post-training outputs and cannot establish an effect with truncation held constant. Controlled prompt or budget interventions, uniform validation of revised scoring, and cross-seed item-level checks remain future work.

The accompanying `data/finding1_audit/` directory preserves the derived QA, mathematics, code, and protocol tables and the sampled case records. The full original audit archive additionally contains raw outputs, evaluator source, and execution logs. This manuscript bundle does not claim to include that complete archive or the two additional seed records.