APPENDIX

A FULL EVALUATION SETUP

A.1 POST-TRAINING METHODS

Reinforcement Learning Reinforcement Learning (RL) has recently proven effective at steering large language models toward complex, multi-step objectives by optimizing policies with scalar reward signals (Zeng et al., 2025). For our experiments, we utilized the `easy-r1` framework, a fork of the original `veRL` project (Yaowei Zheng, 2025). We employed its implementation of the Group Relative Policy Optimization (GRPO) (Shao et al., 2024) algorithm to fine-tune the `Qwen2.5-7B-Instruct` model, using final answer correctness as the reward signal. Our RL configuration uses a learning rate of 1×10^{-6} with the AdamW optimizer and a weight decay of 1.0×10^{-2}. We generate 5 responses per prompt with a maximum total sequence length of 4096 tokens, using a temperature of 1.0 and a top-p of 0.99. The model is updated with a global batch size of 16. KL-divergence regularization was enabled with a coefficient of 1.0×10^{-2}. We trained the model for 5 epochs and selected the checkpoint with the best validation performance.

Supervised Fine-Tuning Supervised Fine-Tuning (SFT) remains a fundamental technique for adapting large pre-trained models by directly minimizing cross-entropy on high-quality datasets (Parashar et al., 2025). We use the LLaMA-Factory framework (Zheng et al., 2024), which is an extensible and user-friendly framework supporting multiple architectures and advanced optimization algorithms, to fine-tune our model on teacher-generated chain-of-thought traces. We use 1×10^{-6} as learning rate, the batch size is 512 and we train for 5 epoch to align with our RL settings.

A.2 DATASETS AND BENCHMARKS

Our analysis was conducted across the following four benchmarks, chosen to cover a range of mathematical and general reasoning tasks:

- **MATH** (Hendrycks et al., 2021): A challenging dataset of 12,500 competition mathematics problems designed to test mathematical problem-solving.
- **GSM8K** (Cobbe et al., 2021): A dataset of 8,500 high-quality, linguistically diverse grade school math word problems created to measure multi-step reasoning.
- **HeadQA** (Vilares & Gómez-Rodríguez, 2019): A multiple-choice question answering dataset sourced from Spanish medical board exams, covering a wide range of topics and requiring specialized knowledge.
- **DeepScaler** (Luo et al.): A proprietary, in-house dataset created to evaluate specific mathematical reasoning abilities. It contains approximately 40,000 unique math problem-answer pairs compiled from sources like the AIME, AMC, Omni-MATH, and Still datasets.

A.3 IMPLEMENTATION DETAILS

All experiments were conducted on a single server equipped with 4 NVIDIA A100 (80GB) GPUs. Our implementation relies on PyTorch and the Hugging Face Transformers library.

B DETAILED DATA FOR DIFFICULTY-STRATIFIED ANALYSIS

B.1 AUTOMATED DIFFICULTY LEVEL ANNOTATION

To ensure a systematic and reproducible partitioning of our datasets into difficulty levels (L1-L5), we employed an automated annotation pipeline. Instead of relying on subjective manual labeling, we developed a detailed rubric based on the cognitive complexity required for each problem and used a large language model (`Gemini 2.5 Pro`) to assign a difficulty score to each problem in our corpus.

The process was guided by the five-level standard defined below. For each problem, the full text of this rubric was provided to the LLM, which was then prompted to return the single most appropriate difficulty level.

Level 1: Direct Application of Basic Rules. Problems that can be solved in one or two steps, where each step is a direct application of a basic formula or operational rule. The solution path is linear and requires minimal strategic planning.

Level 2: Identification of Standard Models. Problems that require identifying the correct standard model or general formula from a set of known methods. This tests for "pattern recognition" of classic problem types.

Level 3: Multi-Step, Cross-Conceptual Planning. Problems that cannot be solved by a single standard model and require a coherent plan that links multiple concepts or steps, often from different mathematical areas.

Level 4: Application of Abstract Concepts. Problems requiring a deep understanding and flexible application of a major, abstract mathematical theory. The solution process is often non-intuitive and relies on a foundational result within a branch of mathematics.

Level 5: Axiomatic Reasoning and Creation. Problems that require reasoning "from first principles" within an axiomatic framework. This involves performing logical deductions, constructing proofs, or finding counterexamples based on the foundational rules of a mathematical structure.

The entire dataset was processed using a parallelized script with a thread pool executor to efficiently query the LLM API. The script included robust error handling and checkpointing to ensure the complete and accurate annotation of the corpus.

B.2 Result

This section provides the full cross-difficulty generalization performance matrices that form the basis for the analysis in Section 3.1 and the visualizations in Figure 2. Table 5 presents the results for the `Qwen2.5-3B-Instruct` model, and Table 6 presents the results for the `Qwen2.5-7B-Instruct` model.

The data in these tables highlights the two key phenomena discussed in the main text. First, the asymmetric generalization is visible by comparing the top-right and bottom-left quadrants of the matrices. For instance, in Table 6, the model trained on Level 5 achieves 94.50% on Level 1, while the model trained on Level 1 only achieves 52.00% on Level 5. Second, the deceptive nature of the average score is evident in the rightmost 'Average' column, where the scores for all five specialist models are remarkably similar (e.g., ranging only from 78.60% to 80.10% for the 7B model), despite their vastly different generalization profiles.

Table 5: Cross-Difficulty Generalization Performance Matrix for the *Qwen2.5-3B-Instruct* model. All values are pass@1 accuracy.

Trained on	**Evaluated on Training Set of Level**					
	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Average**
Level 1	94.50%	85.00%	71.00%	66.00%	41.00%	71.50%
Level 2	93.00%	87.50%	73.00%	65.00%	42.50%	72.20%
Level 3	92.50%	86.00%	75.00%	66.00%	40.00%	71.90%
Level 4	92.50%	86.50%	72.00%	68.00%	43.00%	72.40%
Level 5	94.00%	87.00%	73.00%	62.00%	46.50%	72.50%
Original	92.00%	83.50%	69.50%	62.50%	43.50%	70.20%

Table 6: Cross-Difficulty Generalization Performance Matrix for the *Qwen2.5-7B-Instruct* model. All values are pass@1 accuracy.

Trained on	Evaluated on Training Set of Level					
	Level 1	Level 2	Level 3	Level 4	Level 5	Average
Level 1	97.00%	90.00%	78.00%	76.00%	52.00%	78.60%
Level 2	94.00%	91.50%	82.50%	76.00%	54.00%	79.60%
Level 3	95.50%	91.00%	83.50%	72.50%	56.50%	79.80%
Level 4	93.50%	88.50%	81.00%	80.00%	57.00%	80.00%
Level 5	94.50%	91.00%	78.00%	73.00%	64.00%	80.10%
Original	95.50%	87.50%	76.50%	74.00%	52.00%	77.60%

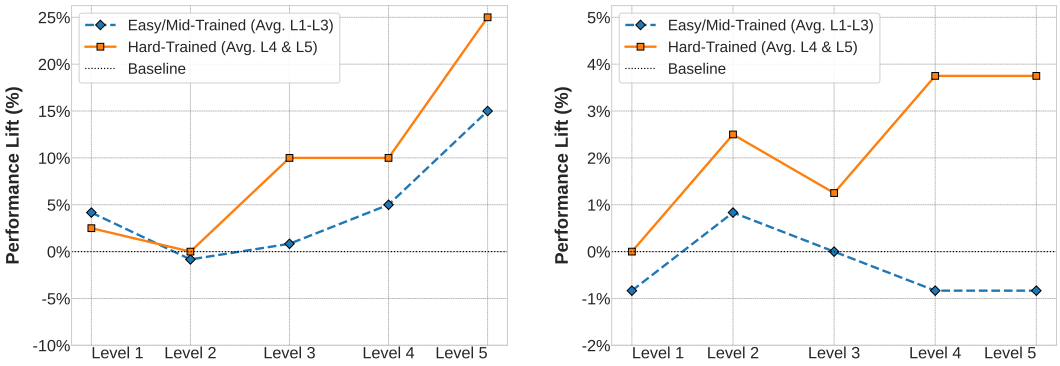
C A Supplementary Experiment to the Difficulty Test

This appendix provides the full performance data for the "generalist-optimized" models described in our supplementary experiment on the difficulty test. The performance lift curves presented in Figure 4 in the main text are directly derived from the raw accuracy scores presented here. Table 7 details the results for the 7B model, while Table 8 shows the results for the 3B model.

Setup. To investigate the impact of training data difficulty on final generalization, we conduct a complexity test. We first train five generalist-optimized models, M_{L_i} for $i \in \{1, \ldots, 5\}$, on the previously defined difficulty-stratified training sets, $\mathcal{D}_{\text{train}}^{L_i}$. The key difference from our prior analysis lies in the evaluation protocol, which is centered around a novel, balanced test set.

- **`Test_Balanced`:** This is the unified and balanced evaluation suite, constructed by sampling an equal number of problems from each of the five difficulty levels. This results in a test set $\mathcal{D}_{bal}$ composed of five equal-sized partitions, $\{\mathcal{D}_{\text{test, bal}}^{L_j}\}_{j=1}^{5}$.

Unlike the models in the first experiment, these models are "generalist-optimized," meaning we select the checkpoint for each M_{L_i} with the highest overall accuracy on the `Test_Balanced` set.



(a) Performance lift of the 3B model.

(b) Performance lift of the 7B model.

Figure 4: *Asymmetric Generalization is consistent across model scales.* Across both the 3B model (a) and the 7B model (b), training on high-difficulty problems (L4-L5, orange line) yields a uniformly superior performance lift over training on easier problems (L1-L3, blue line), proving that mastering complexity is essential for acquiring robust, transferable skills.Full performance data is provided in Table 8 and Table 7.

Our complexity test reveals a stark pattern of asymmetric generalization, as illustrated in Figure 4. Models trained on high-difficulty problems (L4-L5) demonstrate a uniformly superior performance profile, outperforming their counterparts trained on easier data (L1-L3) across all evaluated task complexities. This finding has a critical implication for how we create datasets to train capable models: **the training data must include a significant proportion of difficult problems**. Therefore, for benchmark suites to drive meaningful progress, it is crucial that their provided training sets are sufficiently challenging to promote the development of truly robust models. The data in these

tables clearly illustrates the "asymmetric generalization" phenomenon. For example, in Table 8, the model trained on Level 1 (M_{L_1}) achieves high accuracy (97.50%) on Level 1 test problems but sees its performance drop to just 32.50% on Level 5 problems. In contrast, the model trained on Level 5 (M_{L_5}) maintains robust performance across all levels, demonstrating a more generalizable capability.

Table 7: Performance of *Qwen2.5-7B* generalist-optimized models on the balanced test set. Each row represents a model trained on a specific difficulty level (L_i), evaluated across test questions of all five difficulty levels.

Trained on	**Evaluated on Test Set Questions of Level**					
	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Average**
Level 1	97.50%	90.00%	82.50%	75.00%	50.00%	79.00%
Level 2	95.00%	90.00%	80.00%	77.50%	47.50%	79.00%
Level 3	97.50%	85.00%	85.00%	77.50%	50.00%	79.00%
Level 4	97.50%	87.50%	85.00%	80.00%	55.00%	81.00%
Level 5	97.50%	92.50%	82.50%	82.50%	52.50%	81.50%
Original	97.50%	87.50%	82.50%	77.50%	50.00%	79.00%

Table 8: Performance of *Qwen2.5-3B* generalist-optimized models on the balanced test set. The performance decay for models trained on easy levels (L1, L2) is particularly pronounced.

Trained on	**Evaluated on Test Set Questions of Level**					
	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Average**
Level 1	97.50%	82.50%	75.00%	72.50%	32.50%	72.00%
Level 2	95.00%	87.50%	80.00%	65.00%	35.00%	72.00%
Level 3	97.50%	90.00%	80.00%	72.50%	45.00%	77.00%
Level 4	95.00%	87.50%	87.50%	75.00%	47.50%	78.50%
Level 5	95.00%	87.50%	87.50%	75.00%	47.50%	78.50%
Original	92.50%	87.50%	77.50%	65.00%	22.50%	69.00%

D Data Construction Protocol for the Distribution Test

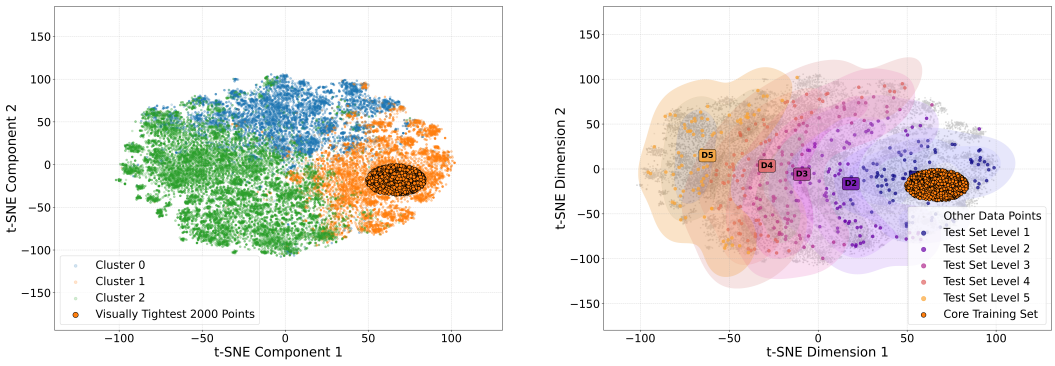
This section details the step-by-step procedure used to construct the specialized training and test sets for the Distribution Test, as described in Section 3.2.1. The entire process is designed to create a controlled environment for measuring generalization as a function of semantic distance. The process consists of three main stages:

Step 1: Semantic Embedding and Clustering. We began with our full corpus of approximately 44785 mathematics problems. To understand their semantic relationships, we first encoded each problem into a high-dimensional vector representation using the `all-mpnet-base-v2` sentence encoder. We then applied K-Means clustering to this high-dimensional embedding space. Using a combination of the Elbow method and Silhouette score analysis, we determined the optimal number of clusters to be $k = 3$, effectively partitioning the entire dataset into three broad, semantically coherent groups.

Step 2: Core Training Set (`Train_Core`) Selection. Our goal was to create a highly concentrated, semantically narrow training set. To achieve this, we first projected the high-dimensional embeddings into a 2D space using t-SNE for visualization. We then focused on a single target cluster (e.g., Cluster 1). Instead of sampling from the high-dimensional space, our selection was based on the *visual density* in the 2D projection. Using the 'NearestNeighbors' algorithm on the 2D t-SNE coordinates, we identified the point within the target cluster whose 2,000 nearest neighbors occupied the smallest possible Euclidean radius. These 2,000 points, representing the most visually compact region of the cluster, formed our exclusive `Train_Core` training set.

Step 3: Distance-Stratified Test Set Construction. To create test sets with increasing semantic distance, we used the remaining 42785 problems not selected for `Train_Core`. First, we calculated the geometric centroid of the 2,000 `Train_Core` points in the 2D t-SNE space. Then, for every other point in the dataset, we computed its Euclidean distance to this centroid. All candidate test points were then sorted based on this distance, from nearest to farthest. This sorted list was partitioned into five equal-sized bins. Finally, we randomly sampled 80 problems from each bin to create our five final test sets, D1 (semantically closest) through D5 (semantically farthest).

The entire data construction pipeline is visually summarized in Figure 5. Panel (a) illustrates the outcome of the `Train_Core` selection process described in Step 2, while Panel (b) shows the resulting distribution of the five distance-stratified test sets as detailed in Step 3.



(a) Training set selection.

(b) Test set distribution.

Figure 5: *Visualization of the experimental data construction for the distribution test.* (a) The highly concentrated $\mathcal{D}_{\text{core}}$ set is selected from a semantic cluster. (b) The test sets are sampled and binned based on their increasing semantic distance from the $\mathcal{D}_{\text{core}}$ centroid.

E THE COUNTERFACTUAL ROBUSTNESS TEST

This section provides detailed, qualitative examples of how fine-tuned models fail on counterfactual reasoning tasks, as discussed in Section 3.2.2. Each table analyzes a specific failure case, comparing the required reasoning path (based on the novel, counterfactual premise) with the model's actual thought process. These examples concretely illustrate the models' strong tendency to disregard explicit instructions and default to their pre-trained, memorized knowledge.

E.1 METHODOLOGY: AUTOMATED DATASET GENERATION

To ensure the diversity and systematic nature of our counterfactual examples, we developed and executed the following automated pipeline, moving beyond manual creation.

Step 1: Strategy — LLM as Data Creator. Our core strategy was to leverage a powerful Large Language Model to act as a creative research assistant. This approach allows for the large-scale and consistent application of complex transformation rules needed to create a high-quality counterfactual dataset.

Step 2:Task Definition — The Counterfactual Transformation. We provided the LLM (`Gemini 2.5 Pro`) with a detailed, multi-step prompt that precisely defined the transformation task. The instructions guided the model to first analyze a given standard problem to identify a core logical or mathematical rule. Subsequently, the model was tasked to invent a plausible but contrary-to-fact rule, rewrite the problem statement to include this new premise, and finally, generate a new step-by-step solution based exclusively on the novel rule.

Step 3: Execution — Parallelized Pipeline. This generation process was applied to our entire source dataset. To manage the scale, the pipeline was executed in parallel using a Python script with a `ThreadPoolExecutor` to handle concurrent API requests. The full, unabridged master prompt used in this process is available in our supplementary materials to ensure full reproducibility.

E.2 Case Study: Arithmetic Order of Operations

(Counterfactual Premise)

A novel order of operations, **PESAMD**, is defined: Parentheses, Exponents, **S/A**, then **M/D**. The model is asked to evaluate $f(x) = \frac{3x-2}{x-2}$.

Correct Reasoning (PESAMD)
1. **Numerator (S first):** $3 \times (0-2) = -6$
2. **Denominator:** $0 - 2 = -2$
3. **Division (last):** $\frac{-6}{-2} = 3$

The final correct answer is $\mathbf{9}$.

Model's Actual Reasoning
1. **Numerator (M first):** It computes $3 \times 0 = 0$ first, then $0 - 2 = -2$. This follows the **memorized PEMDAS rule**, violating the premise.
2. **Denominator:** Correctly computes $0 - 2 = -2$.
3. **Division:** $\frac{-2}{-2} = 1$.

The final incorrect answer is $\mathbf{\frac{14}{3}}$.

E.3 Case Study: Number Theory Divisor Rule

(Counterfactual Premise)

A new system defines the number of divisors of $N = p_1^{a_1} \cdots$ as the **sum** of $(a_i + 1)$ values. Find the number of divisors for $N = 12$.

Correct Reasoning (Sum Rule)
1. Prime factorization of 12 is $2^2 \times 3^1$.
2. The exponents are $a_1 = 2, a_2 = 1$.
3. Apply the new **sum rule**: $(2+1)+(1+1) = 5$.

The final correct answer is $\mathbf{5}$.

Model's Actual Reasoning
1. Correctly finds prime factorization: $12 = 2^2 \times 3^1$.
2. Ignores the "sum" rule and applies the memorized "product" rule: $(2+1) \times (1+1) = 6$.

The final incorrect answer is $\mathbf{6}$.

E.4 Case Study: Physics Speed Formula

(Counterfactual Premise)

A car travels 120 km in 2 hours. In this reality, 'average speed' is calculated as: **speed = time / distance**. Find the speed.

Correct Reasoning (New Formula)
1. Identify Time = 2 hours, Distance = 120 km.
2. Apply the new formula time / distance: $2 \div 120 = \frac{1}{60}$.

The final correct answer is $\mathbf{\frac{1}{60}}$ **km/h**.

Model's Actual Reasoning
1. Correctly identifies Time and Distance.
2. Ignores the new formula and applies the memorized, standard formula 'distance / time': $120 \div 2 = 60$.

The final incorrect answer is **60 km/h**.

F USAGE OF LARGE LANGUAGE MODELS

We declare that LLMs were used solely for language polishing purposes in this work. Specifically, after completing the initial draft entirely through human effort, we employed LLM assistance exclusively for grammatical refinement and improving the clarity of English expression to meet academic writing standards. All intellectual contributions, from conceptualization to initial manuscript preparation, were performed by the human authors. The use of LLM was limited to post-writing language enhancement, similar to traditional proofreading services, ensuring that non-native English speakers can present their research with appropriate linguistic quality while maintaining complete authorship and originality of the scientific content.