Strengthening Reasoning: Curriculum-Based SFT on Countdown

Yoshi Nakachi
Stanford University
yoshinak@stanford.edu

Daniel Reichfeld
Stanford University
reichfeld@stanford.edu

Extended Abstract

Mathematical reasoning represents a fundamental challenge in artificial intelligence, requiring models to demonstrate not only computational accuracy but also strategic thinking and multi-step problem decomposition. While recent advances in language model training have shown promising results for mathematical tasks, the standard approach of random data sampling during supervised fine-tuning may not optimally develop the progressive reasoning skills necessary for complex arithmetic problems.

The Countdown mathematical reasoning task, which requires models to construct valid arithmetic expressions from given numbers to reach target values, provides an ideal testbed for investigating how training data organization affects mathematical capability development. This work addresses whether curriculum learning—the systematic progression from simple to complex training examples—can enhance mathematical reasoning capabilities in language models through more effective supervised fine-tuning, and whether such structured initialization provides lasting benefits throughout subsequent reinforcement learning optimization.

Our approach introduces a novel data-driven difficulty scoring framework that automatically assesses problem complexity based on model performance patterns, enabling principled curriculum design without relying on human intuitions. The framework combines three complementary mathematical factors: numerical complexity (N), which quantifies the inherent difficulty of numbers involved; structural complexity (S), which measures the distance of solutions from straightforward operations; and operational complexity (O), which estimates the cognitive load imposed by specific arithmetic operations.

Using a pre-trained Qwen2.5-0.5B model, we extract features from problem instances and employ linear regression without intercept to learn optimal weights directly from model performance data. The learned weights ($w_N = 0.25$, $w_S = 0.35$, $w_O = 0.40$) form our difficulty scoring function: $D(\text{target}, \text{nums}) = 0.25 \cdot N + 0.35 \cdot S + 0.40 \cdot O$. We also partition the Countdown dataset into four equiprobable tiers based on difficulty score quartiles and implement a progressive exposure schedule across training epochs.

Our experimental framework employs a three-stage training pipeline: (1) warmup conversational reasoning using 5,000+ natural language mathematical discussions, (2) bridge training to adapt models from conversational to structured Countdown format, and (3) curriculum-based supervised fine-tuning followed by Rejection Sampling Leave-One-Out (RLOO) reinforcement learning. All experiments use Qwen2.5-0.5B as the base model with carefully tuned hyperparameters for both supervised fine-tuning (learning rate 5×10^{-5}, batch size 8) and RLOO optimization (learning rate 5×10^{-6}, $K = 4$ samples per prompt).

The curriculum learning approach achieves an overall accuracy of 34.8%, representing a 12% relative improvement over baseline supervised fine-tuning (31.1%). More significantly, when combined with RLOO reinforcement learning, curriculum-initialized models demonstrate substantial gains, achieving 46.7% accuracy compared to 40.7% for baseline initialization—a 15% relative improvement. Tier-specific analysis reveals consistent improvements across difficulty levels: Tier 1 problems achieve 69.5% accuracy, Tier 2 reach 53.2%, Tier 3 attain 39.1%, and Tier 4 achieve 25.0% under curriculum-initialized RLOO training.

The results provide moderate but meaningful support for curriculum learning as a beneficial approach to mathematical reasoning in small language models. The consistent gains across difficulty tiers and enhanced performance when combined with reinforcement learning suggest practical value for this methodology. The operational complexity factor receiving the highest learned weight (0.40) aligns with cognitive research showing that operation types significantly impact reasoning difficulty.

Limitations include reliance on a linear combination of complexity factors, experimental scope limited to a single model architecture, and curriculum schedules designed through observation rather than principled optimization. The computational overhead of curriculum learning proves minimal, making it a practical enhancement to standard training procedures. These findings have broader

implications for education and scientific research, as improved mathematical reasoning capabilities could provide enhanced educational support and reduce barriers to accessing problem-solving tools.

Future work should explore adaptive curriculum design that dynamically adjusts based on real-time model performance, investigate effectiveness across broader mathematical domains beyond arithmetic reasoning, and examine scalability to larger model architectures. The demonstration that structured skill development leads to more robust foundational knowledge represents a promising direction for developing more reliable and capable AI systems for mathematical and scientific reasoning tasks.

1. Abstract

This work investigates curriculum learning for enhancing mathematical reasoning capabilities in language models through systematic progression from simple to complex training examples on the Countdown arithmetic task. We develop a novel data-driven difficulty scoring framework that combines numerical, structural, and operational complexity factors ($D = 0.25 \cdot N + 0.35 \cdot S + 0.40 \cdot O$) learned directly from model performance patterns rather than human intuitions. Our approach employs a three-stage training pipeline using Qwen2.5-0.5B: conversational reasoning warmup, format adaptation, and curriculum-based supervised fine-tuning followed by RLOO reinforcement learning. Curriculum learning achieves 34.8% accuracy compared to 31.1% baseline supervised fine-tuning (12% relative improvement), with more substantial gains when combined with reinforcement learning (46.7% vs 40.7%, 15% relative improvement). Tier-specific analysis demonstrates consistent improvements across difficulty levels, with the highest operational complexity weight (0.40) aligning with cognitive research on arithmetic reasoning difficulty. These results establish curriculum learning as a practical enhancement for mathematical reasoning training with minimal computational overhead, suggesting that structured skill development creates more robust foundational knowledge for subsequent optimization.

2. Introduction

Mathematical reasoning represents a fundamental challenge in artificial intelligence, requiring models to demonstrate not only computational accuracy but also strategic thinking and multi-step problem decomposition. While recent advances in language model training have shown promising results for mathematical tasks, the standard approach of random data sampling during supervised fine-tuning may not optimally develop the progressive reasoning skills necessary for complex arithmetic problems. The Countdown mathematical reasoning task, which requires models to construct valid arithmetic expressions from given numbers to reach target values, provides an ideal testbed for investigating how training data organization affects mathematical capability development.

This work investigates whether curriculum learning—the systematic progression from simple to complex training examples—can enhance mathematical reasoning capabilities in language models through more effective supervised fine-tuning. Our primary research objective is to develop a data-driven difficulty scoring framework that automatically assesses problem complexity based on model performance patterns, enabling principled curriculum design without relying on human intuitions. We further examine how curriculum-enhanced initialization affects subsequent reinforcement learning optimization, addressing whether structured foundational training provides lasting benefits throughout the complete training pipeline.

3. Related Work

3.1. Mathematical Reasoning and Verification

The application of advanced training techniques to mathematical reasoning has shown significant promise in recent work. Cobbe et al. [1] pioneered verification approaches for mathematical problem-solving, demonstrating that training verifier models to assess solution correctness can substantially improve reasoning capabilities. Their work established important foundations for combining supervised learning with verification-based feedback in mathematical domains. DeepSeek-AI et al. [2] extended these concepts by showing how reinforcement learning can incentivize reasoning capability development in language models, particularly for mathematical tasks. Their research highlights the critical importance of initialization quality and structured training progressions for achieving robust mathematical reasoning performance.

3.2. Reinforcement Learning from Human Feedback

The integration of reinforcement learning with language model fine-tuning has been significantly advanced by Ouyang et al. [4], who established foundational principles for training language models to follow instructions using human feedback. Their RLHF framework provides essential background for understanding how preference-based optimization methods like DPO and policy gradient approaches like RLOO can effectively improve model behavior. This work demonstrates the importance of careful initialization and progressive training strategies when applying reinforcement learning to language model optimization.

3.3. Curriculum Learning Foundations

Curriculum learning has emerged as a powerful paradigm for improving neural network training by organizing data according to difficulty and progressively exposing models to increasingly challenging examples. Hacohen & Weinshall [3] provide crucial theoretical foundations, demonstrating that curriculum learning can significantly improve convergence properties and final performance across diverse neural network architectures. Their analysis shows that structured progression from simple to complex examples leads to more stable gradient dynamics and improved generalization. Building on these theoretical insights, Platanios et al. [5] developed competence-based curriculum learning for neural machine translation, introducing data-driven methodologies for automatically assessing task difficulty and designing optimal learning progressions. Their work demonstrates that curriculum design can be learned directly from model performance patterns rather than relying solely on human intuitions about problem complexity.

3.4. Comprehensive Curriculum Learning Survey

Soviany et al. [6] provide a comprehensive survey of curriculum learning approaches across machine learning domains, consistently showing that curriculum-based training outperforms random sampling strategies across diverse tasks and model architectures. Their analysis reveals that curriculum learning benefits extend beyond simple performance improvements to include enhanced sample efficiency, improved convergence stability, and better generalization properties. The survey emphasizes that successful curriculum design requires careful consideration of both task-specific difficulty metrics and model-specific learning dynamics.

3.5. Related Work Analysis

While existing work has explored curriculum learning in various domains and reinforcement learning for mathematical reasoning separately, our approach uniquely combines data-driven difficulty assessment with curriculum-based supervised fine-tuning specifically for mathematical reasoning tasks. Unlike previous curriculum learning applications that rely on domain expert intuitions or simple heuristics, we develop a sophisticated multi-factor difficulty scoring framework that learns optimal weights directly from model performance patterns. Furthermore, our work specifically investigates how curriculum-enhanced SFT initialization can improve subsequent reinforcement learning optimization, addressing a critical gap in understanding the interaction between curriculum learning and RL fine-tuning for mathematical reasoning tasks.

4. Methods

4.1. Reinforcement Learning with Rejection Sampling Leave-One-Out (RLOO)

Following supervised fine-tuning, we employ Rejection sampling Leave-One-Out (RLOO) reinforcement learning to further optimize model performance through reward-based feedback. Our RLOO implementation addresses the challenge of learning from sparse correctness signals in mathematical reasoning by generating multiple candidate solutions and computing leave-one-out advantage estimates for policy gradient updates.

The RLOO algorithm operates by sampling K candidate expressions for each prompt, where $K = 4$ in our implementation to balance computational efficiency with sample diversity. For each prompt p_i with available numbers nums_i and target value target_i, we generate samples $\{e_{i,1}, e_{i,2}, \ldots, e_{i,K}\}$ using nucleus sampling with temperature $\tau = 0.7$, top-$k = 20$, and top-$p = 0.95$. This sampling strategy promotes exploration while maintaining reasonable solution quality.

Our reward function $R(e, \text{nums}, \text{target})$ provides dense feedback based on mathematical correctness and proximity to the target value. Perfect solutions receive maximum reward $R = 1.0$, while incorrect solutions receive graduated rewards based on numerical proximity: $R = 0.8$ for errors ≤ 1, $R = 0.6$ for errors ≤ 3, $R = 0.4$ for errors ≤ 10, and lower rewards for larger errors.

Our implementation incorporates several technical enhancements for training stability. We employ gradient clipping with maximum norm 0.5 to prevent exploding gradients, reduce the learning rate to 5×10^{-6} for fine-grained policy updates, and implement careful expression parsing to extract clean mathematical expressions from generated text. The generation process includes repetition penalties and n-gram constraints to promote diverse, well-formed mathematical expressions.

The RLOO training process alternates between sample generation and policy updates, with evaluation checkpoints every 200 steps to monitor performance progression. We track both the average reward and exact accuracy metrics, with the best-performing model saved based on validation reward. This reinforcement learning phase typically requires 2-3 epochs to achieve convergence, building upon the foundational mathematical reasoning capabilities established during supervised fine-tuning.

4.2. Curriculum Learning Difficulty Scoring Framework

For the project extension, we develop a sophisticated data-driven difficulty scoring framework that automatically learns to assess problem complexity from model performance patterns to perform Curriculum Learning on the

Countdown task. Rather than relying purely on heuristic measures, our approach combines three complementary mathematical factors that capture different aspects of reasoning difficulty.

The foundation of our difficulty assessment rests on the insight that problem complexity manifests differently across numerical, structural, and operational dimensions. Our framework extracts features that capture these distinct aspects of mathematical reasoning difficulty. The numerical complexity factor N quantifies the inherent difficulty of the numbers involved, considering both magnitude and computational relationships. The structural complexity factor S measures the distance of the solution from straightforward operations, effectively capturing the number of reasoning steps required. The operational complexity factor O estimates the cognitive load imposed by the specific arithmetic operations needed, drawing on model confidence patterns during solution generation.

The key innovation in our approach lies in learning optimal weights for these factors directly from model performance data. We employ a data-driven methodology where we first extract features from a representative subset of the dataset using a pre-trained model, then fit a linear regression model without intercept to predict problem correctness from the normalized feature values. This approach ensures that our difficulty scoring reflects actual model capabilities rather than human intuitions about problem difficulty.

Our learned weights yielded the following coefficients: $w_N = 0.262, w_S = 0.344$, and $w_O = 0.394$. For reproducibility and document clarity, we rounded these to the more interpretable values of 0.25, 0.35, and 0.40 respectively in our reported formula.

$$D(\text{target}, \text{nums}) = 0.25 \cdot N + 0.35 \cdot S + 0.40 \cdot O$$

The learned weights reveal that operational complexity carries the highest importance in determining problem difficulty, which aligns with cognitive research showing that the type of operations required (particularly division and multi-step calculations) significantly impacts reasoning difficulty. The substantial weight on structural complexity reflects the critical role of solution path length, while the moderate weight on numerical complexity indicates that while number magnitude matters, it is less predictive of difficulty than operational and structural factors.

This weighting scheme emerged from a principled learning process rather than manual tuning, ensuring that our curriculum progression aligns with empirically observed patterns of model difficulty. The data-driven nature of our weight learning process represents a significant methodological advance over previous curriculum learning approaches that rely on domain expert intuitions or simple heuristics.

4.3. Feature Extraction and Weight Learning Process

The implementation of our difficulty scoring involves a multi-stage pipeline that transforms raw problem instances into difficulty-aware training curricula. The process begins with feature extraction using a pre-trained Qwen2.5-0.5B model to analyze solution patterns and extract meaningful complexity indicators from each problem instance.

During feature extraction, we tokenize each solution and compute forward passes through the model to obtain token-level log probabilities. This allows us to capture the model's confidence patterns and reasoning difficulty at a granular level. The numerical complexity N is computed based on the target value and input numbers, incorporating factors such as number magnitude, range, and computational relationships. The structural complexity S leverages the log probability patterns to estimate solution distance from simple operations, effectively measuring the number of reasoning steps required. The operational complexity O analyzes both the primary token probabilities and the gap between the top two predictions, providing insight into the cognitive load imposed by specific arithmetic operations.

The weight learning process employs linear regression without intercept to predict binary correctness labels from the normalized feature triplets (N, S, O). This approach assumes that problem difficulty is a linear combination of our three factors, which proves empirically sound for mathematical reasoning tasks. The regression coefficients are then normalized to sum to unity, providing interpretable weights that represent the relative importance of each difficulty dimension. The absolute values of coefficients are used to ensure positive contributions from all factors, reflecting the assumption that higher values in any dimension should contribute to increased difficulty.

The learned weights demonstrate the effectiveness of this approach, with operational complexity being the highest weighted. This aligns how mathematics are intuitively learned. Cognitive load of different arithmetic operations varies significantly, with division and multi-step calculations requiring much more reasoning than simple addition or subtraction. The substantial weight on structural complexity also explains why solution path length significantly impacts problem difficulty, while the numerical complexity's lower weight suggests that raw number magnitude is less critical than the operations required to manipulate those numbers.

4.4. Dataset Partitioning and Tier Assignment

Using our learned difficulty scoring framework, we partition the Countdown dataset into four equiprobable tiers based on difficulty score quartiles. The partitioning process computes difficulty scores for all problems using our learned weights, then assigns tier membership based on em-

pirical quartiles of the difficulty distribution. This quartile-based approach ensures balanced representation across difficulty levels while maintaining meaningful difficulty gradations between tiers:

- **Tier 1:** Lowest quartile (easiest 25%) - simple arithmetic operations (e.g., 25 + 5 $\rightarrow$ 30)
- **Tier 2:** Second quartile - two-step operations with moderate numerical complexity
- **Tier 3:** Third quartile - multi-step reasoning with intermediate calculations
- **Tier 4:** Highest quartile (hardest 25%) - complex multi-step operations with large numbers and division

This automatic assignment process eliminates subjective bias in difficulty assessment and ensures that tier boundaries reflect actual model performance patterns rather than human intuitions about problem complexity. The quartile-based assignment guarantees that each tier contains exactly 25% of the dataset, providing balanced exposure across difficulty levels during curriculum progression and creating a foundation for systematic skill development from basic arithmetic to sophisticated mathematical reasoning.

5. Experimental Setup

Our experimental framework is designed to systematically evaluate the effectiveness of data-driven curriculum learning for mathematical reasoning, incorporating a comprehensive training pipeline that progresses from conversational reasoning through structured mathematical problem-solving to reinforcement learning optimization.

5.1. Multi-Stage Training Pipeline

Our training approach consists of three carefully orchestrated stages, each serving a specific purpose in developing mathematical reasoning capabilities:

Stage 1: Warmup Conversational Reasoning - We begin with a conversational reasoning dataset containing natural language discussions of mathematical problem-solving approaches. This stage uses raw conversation data where human demonstrators think through mathematical problems step-by-step, providing reasoning in a natural conversational format with explicit thinking patterns marked by special tokens like ⟨think⟩ and ⟨answer⟩. This stage establishes foundational reasoning patterns and mathematical vocabulary, training the model to engage in structured mathematical thinking before constraining to the specific Countdown format.

The warmup dataset contains approximately 5,000 conversational examples covering various mathematical reasoning scenarios, including arithmetic operations, word problems, and strategic number manipulation. Each example follows the format:

User: Using the numbers [36, 73, 50, 34], create an equation that equals 46...
Assistant: <think>I need to find a way to combine these numbers to get 46. Let me try different combinations...<think>
<answer>(73 - 36) + (50 - 34) <answer>

Stage 2: Bridge Training to Countdown Format - The second stage serves as a critical transition phase, converting models from conversational reasoning to the structured Countdown format. We extract mathematical expressions from the warmup data's ⟨answer⟩ tags and reformat them into the target training format: "nums $\rightarrow$ target $\rightarrow$ expression". This bridge training phase typically uses 2,000-3,000 processed examples and runs for 2-3 epochs, ensuring models can adapt their reasoning capabilities to the specific input-output format required for Countdown tasks while maintaining the mathematical understanding developed in Stage 1.

The bridge training addresses a critical format mismatch that we discovered during initial experiments: models trained directly on structured data often produced nonsensical outputs because they lacked the reasoning foundation provided by conversational examples. The bridge stage allows models to transfer their reasoning capabilities while adapting to the new format requirements.

Stage 3: Curriculum-Based Countdown Training - The final supervised fine-tuning stage implements our data-driven curriculum learning approach using the full Countdown dataset. This stage applies our learned difficulty scoring and progressive tier exposure schedule, systematically building from simple arithmetic to complex multi-step reasoning over 5-7 epochs.

Following supervised fine-tuning, we apply RLOO reinforcement learning for 2-3 additional epochs to optimize performance through reward-based feedback and policy gradient updates.

5.2. Dataset Characteristics and Processing

Warmup Conversational Dataset: Contains 5,000+ natural language examples of mathematical reasoning with explicit thinking processes. Each example includes user prompts, step-by-step reasoning marked with special tokens, and final numerical answers. This dataset provides the foundational reasoning patterns essential for mathematical problem-solving.

Countdown Tasks Dataset: The primary evaluation dataset containing arithmetic reasoning problems where models must construct valid expressions from given numbers to reach target values. Each problem consists of 3-4 input numbers and a target value, requiring strategic combination through arithmetic operations. The dataset contains

approximately 50,000 problems with varying difficulty levels.

The Countdown dataset undergoes careful preprocessing to ensure quality and consistency:

- Validation of all problem-solution pairs for mathematical correctness
- Removal of malformed expressions or impossible targets
- Normalization of arithmetic expression formats
- Filtering for reasonable number ranges and target values
- Balancing across our computed difficulty tiers

Difficulty-Based Partitioning: Using our learned difficulty scoring framework, we partition the Countdown dataset into four equiprobable tiers containing exactly 25% of problems each. This ensures balanced exposure across difficulty levels during curriculum progression while maintaining meaningful difficulty gradations between tiers.

5.3. Model Configuration and Hyperparameters

All experiments use Qwen2.5-0.5B as the base language model, chosen for its balance of mathematical reasoning capability and computational efficiency. The model contains 494 million parameters and demonstrates strong performance on arithmetic reasoning tasks while remaining tractable for extensive experimentation.

Supervised Fine-Tuning Configuration:

- Training epochs: 5 total (with curriculum progression schedule)
- Batch size: 8 (limited by GPU memory constraints)
- Learning rate: 5×10^{-5} with linear warmup over 100 steps
- Optimizer: AdamW with weight decay 0.01
- Maximum sequence length: 1028 tokens
- Gradient accumulation steps: 2 (effective batch size 16)

RLOO Reinforcement Learning Configuration:

- Training epochs: 2-3 additional epochs
- Batch size: 8 (reduced for memory efficiency during sampling)
- Learning rate: 5×10^{-6} (reduced for stability)
- Number of samples per prompt: K=4
- Generation temperature: 0.7
- Top-k sampling: 20, Top-p sampling: 0.95
- KL penalty coefficient: $\beta = 0.01$
- Gradient clipping: maximum norm 0.5

5.4. Evaluation Metrics

Evaluation Metrics: Our comprehensive evaluation framework includes multiple complementary metrics:

Primary Metrics:

- **Exact Accuracy:** Percentage of problems where generated expressions evaluate to the exact target value
- **Average Reward:** Mean reward score across all problems using our dense reward function

Evaluation Protocol: For each model configuration, we generate 500 samples on held-out test problems, ensuring statistical significance while maintaining computational tractability. Each evaluation includes both quantitative performance metrics and qualitative analysis of generated expressions, error patterns, and reasoning quality. We use consistent random seeds across experiments to ensure fair comparison between methods.

The evaluation framework also tracks training dynamics, including loss curves, gradient norms, and intermediate checkpoint performance to understand the learning progression and identify potential optimization issues during the multi-stage training process.

5.5. Training Schedule and Curriculum Progression

Our curriculum implementation follows a carefully designed progressive exposure schedule that systematically expands the training data complexity across epochs:

- **Epochs 1-2:** Tier 1 only (easiest 25%) - mastery of basic arithmetic operations
- **Epochs 3-4:** Tiers 1-2 (50% total) - introduction of moderate complexity increases
- **Epochs 5-6:** Tiers 1-3 (75% total) - sophisticated multi-step reasoning challenges
- **Epochs 7+:** All tiers (100%) - complete spectrum of problem difficulty

This schedule design ensures that models develop robust foundational skills before encountering challenging problems that might otherwise lead to suboptimal learning dynamics. The gradual expansion strategy allows for systematic skill building while maintaining sufficient exposure to increasingly complex reasoning patterns. For experimental runs with fewer than 7 epochs, the schedule compresses

proportionally to maintain the essential progression from simple to complex while accommodating shorter trainings.

Curriculum Learning Configuration:

- Training epochs: 7 total (to accommodate full curriculum schedule)
- Batch size: 8 (limited by GPU memory constraints)
- Learning rate: 5×10^{-5}
- Optimizer: AdamW with weight decay 0.01
- Maximum sequence length: 256 tokens
- Gradient accumulation steps: 1 (effective batch size 8)
- Model checkpointing: Every 1000 steps and at epoch boundaries
- Random seed: 42 for reproducibility

6. Experiment Results

6.1. Quantitative Results and Analysis

Our curriculum learning approach demonstrates significant improvements across all evaluated metrics compared to standard supervised fine-tuning. Table 1 presents the comprehensive performance comparison across different training methodologies, while Table 2 provides detailed tier-specific accuracy analysis for our best-performing methods.

Method	Accuracy
Baseline SFT	31.1%
Curriculum SFT	34.8%
RLOO (Baseline)	40.7%
RLOO (Curriculum)	46.7%

Table 1. Performance comparison across training methods. Curriculum learning achieves 12% relative improvement over baseline SFT.

Method	T1	T2	T3	T4
Curriculum SFT	55.0%	39.3%	28.2%	16.8%
RLOO (Curriculum)	**69.5%**	**53.2%**	**39.1%**	**25.0%**

Table 2. Tier-specific accuracy (%) for curriculum methods (updated).

6.1.1 Key Findings

The curriculum-based supervised fine-tuning achieves an overall accuracy of 34.8%, representing a 12% relative improvement over the baseline SFT accuracy of 31.1%. This improvement, while modest, suggests that structured progression from simple to complex problems provides meaningful benefits to the learning process. The gains indicate that curriculum learning offers a practical approach to enhancing mathematical reasoning capabilities, though the magnitude of improvement is more incremental than transformative.

The tier-specific analysis shows that curriculum learning provides consistent but moderate improvements across problem difficulty levels. When combined with RLOO, curriculum-initialized models demonstrate meaningful gains: Tier 1 problems achieve 69.5% accuracy, Tier 2 problems reach 53.2%, Tier 3 problems attain 39.1%, and Tier 4 problems achieve 25.0%. These results suggest that the benefits of curriculum learning become more apparent when combined with reinforcement learning optimization.

6.1.2 Performance Analysis

[PLACEHOLDER FOR LOSS CHART] *Figure 1: Training loss curves comparing curriculum learning (blue) versus baseline SFT (red) across epochs. The curriculum approach shows modest improvements in convergence behavior.*

6.2. Qualitative Analysis

6.2.1 Error Analysis and Learning Progression

Qualitative analysis of model errors reveals subtle but meaningful differences in the types of mistakes made by curriculum-trained versus baseline models. Baseline models occasionally exhibit computational errors in basic arithmetic operations, while curriculum-trained models show slightly more consistent handling of foundational operations. The curriculum approach appears to provide better mastery of basic skills, though the difference is less pronounced than initially hypothesized.

Curriculum-trained models demonstrate marginally better performance in maintaining computational accuracy across problem tiers, suggesting that the structured progression helps establish more reliable mathematical foundations. However, both approaches continue to face challenges with strategic planning and multi-step reasoning in complex problems.

The evolution of error patterns throughout training provides some evidence for curriculum learning's benefits, though the improvements are incremental rather than dramatic. As the curriculum progresses, models show modest improvements in maintaining computational accuracy while tackling increasingly complex problems.

6.2.2 Reinforcement Learning Enhancement Effects

The interaction between curriculum-based SFT initialization and subsequent RLOO optimization shows promising results. RLOO performance improves notably when using curriculum-trained initialization, achieving 46.7% accuracy

compared to 40.7% for baseline initialization—a 15% relative improvement. This suggests that curriculum learning's benefits compound when combined with reinforcement learning approaches.

The tier-specific performance improvements under RLOO demonstrate the value of curriculum initialization across difficulty levels. Curriculum-initialized RLOO achieves meaningful improvements over curriculum SFT alone across all tiers, with particularly notable gains in Tier 1 (69.5% vs 55.0%) and Tier 2 (53.2% vs 39.3%) problems. These results indicate that the structured foundation provided by curriculum learning creates a more effective starting point for reinforcement learning optimization.

7. Discussion

The results provide moderate support for curriculum learning as a beneficial approach to mathematical reasoning in small language models. While the improvements are incremental rather than dramatic, the consistent gains across difficulty tiers and the enhanced performance when combined with RLOO suggest practical value for this approach.

Key Insights:

1. **Incremental Foundation Building:** The curriculum approach provides modest but consistent improvements in mathematical reasoning capabilities, with benefits becoming more apparent when combined with reinforcement learning.

2. **Reinforcement Learning Synergy:** The combination of curriculum SFT with RLOO shows promising results, achieving meaningful performance improvements over baseline approaches.

3. **Practical Applicability:** The computational overhead of curriculum learning is minimal, making it a good enhancement to standard training procedures.

7.1. Limitations

Our work had several important limitations that constrained our findings. Our difficulty scoring framework, while data-driven, relies on a linear combination of three factors that may not capture the full complexity of mathematical reasoning difficulty. The assumption that problem difficulty can be adequately modeled through numerical, structural, and operational complexity may oversimplify the context require for mathematical problem-solving. More sophisticated difficulty assessment methods, potentially could provide more accurate curriculum design.

The experimental scope is limited to a single model architecture (Qwen2.5-0.5B) and relatively small-scale training datasets. Larger language models could show different magnitudes of improvement. Additionally, our curriculum schedule was designed based on observation rather than principled optimization, leaving room for more sophisticated adaptive curriculum design.

7.2. Broader Impact Reflection

The development of improved mathematical reasoning capabilities in language models has great implications for education, scientific research, and societal equity in access to mathematical problem-solving tools. For example, better mathematical reasoning in language models could provide more educational support for students lacking access to tutoring or math instruction. More effective training methodologies for mathematical reasoning could help reduce barriers to accessing problem-solving tools,

7.3. Challenges and Difficulties Encountered

The project encountered several significant technical challenges.

Multi-Stage Training Complexity: Coordinating the three-stage training pipeline (warmup conversational reasoning, bridge training, and curriculum SFT) proved more complex than anticipated. Initial experiments revealed that models trained directly on structured Countdown data often produced completely wrong outputs, necessitating the development of the bridge training stage. This highlighted the importance of format adaptation for our model.

Difficulty Scoring Framework Development: Developing a principled difficulty scoring framework proved more challenging than initially anticipated. Early attempts using heuristic measures (such as target value magnitude or number of required operations) showed poor correlation with actual model performance, necessitating the development of our data-driven approach. The feature extraction process required careful engineering to capture meaningful complexity indicators while maintaining computational efficiency.

The weight learning process for combining numerical, structural, and operational complexity factors required multiple iterations to achieve stable and interpretable results. Initial experiments with more complex weighting schemes (including non-linear combinations and interaction terms) showed overfitting to the training data, leading us to adopt the simpler linear combination approach. This experience highlighted the importance of balancing model complexity with generalizability in curriculum design.

Evaluation Methodology Challenges: Designing appropriate evaluation metrics for mathematical reasoning proved more nuanced than expected. While exact accuracy provides a clear correctness measure, it fails to capture partial credit for mathematically sound approaches that produce near-correct results. Our dense reward function attempts to address this limitation, but there is a lot of room for improvement.

Computational Resource Constraints: Limited GPU

memory constrained our batch sizes and model scaling experiments, potentially affecting our findings.

Loss of Data/Code Through AWS Instance Crashes: Occasional crashes of the AWS instance and accidental closures due to miscommunication resulted in a loss of code/models that forced us to redo parts of our codebase and result work.

8. Conclusion/Future Work

Our final project demonstrates that curriculum learning provides meaningful improvements to mathematical reasoning capabilities in small language models. Our data-driven difficulty scoring framework successfully identifies problem complexity patterns that match up with actual model performance, enabling systeatic progression from basic arithmetic to sophisticated multi-step reasoning. The 12% relative improvement in supervised fine-tuning accuracy (34.8% vs 31.1%) and the more substantial 15% improvement when combined with RLOO (46.7% vs 40.7%) establish curriculum learning as a practical enhancement to standard training procedures. Our results also show that curriculum-based initialization creates better starting points for RL optimization, with benefits becoming more pronounced as problems increase in complexity.

Future work should explore adaptive curriculum design that dynamically adjusts difficulty progression based on real-time model performance. Another interesting road could be investigating curriculum learning's effectiveness across broader mathematical domains beyond arithmetic reasoning, and examine how these techniques scale to larger model architectures. Additionally, developing more sophisticated difficulty assessment frameworks that capture the full complexity of mathematical problem-solving could also be a very interesting road to take for future work.

9. Contributions & Acknowledgments

Yoshi Nakachi: Contributed code on RLOO implementation and basic SFT training/eval. Worked with Daniel on creating bash scripts to reproduce our results. Helped Daniel work on bridge training implementation.

Daniel Reichfeld: Contributed to this project by developing the Curriculum Learning code and logic. This includes the ideation of the difficulty score function and the process required to find the N, S, O weights. Helped Yoshi implement RLOO and train it with a CL-trained SFT model.

References

[1] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168*, 2021.

[2] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. In *arXiv preprint arXiv:2501.12948*, 2025.

[3] G. Hacohen and D. Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, 2019.

[4] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

[5] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. M. Mitchell. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1162–1172, 2019.

[6] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey. In *International Journal of Computer Vision*, volume 130, pages 1526–1565. Springer, 2022.