The Valley of Code Reasoning: Scaling Knowledge Distillation of Large Language Models

Muyu He **Muhammad Ali Shafique** **Anand Kumar** **Tsach Mackey** **Nazneen Rajani**
Collinear AI
research@collinear.ai

Abstract

Distilling the thinking traces of a Large Language Model (LLM) with reasoning capabilities into a smaller model has been proven effective. Yet, there is a scarcity of work done on how model performances scale with the quantity of distillation data. In this work, we study the scaling trend of distilling competitive coding skills on two small non-reasoning LLMs. We validate the hypothesis that there is a valley of code reasoning: downstream performance on competitive coding first drops as data quantity increases, then it steadily increases in a log-linear fashion. Having identified the trend, we further fine-tune the models at two different distillation stages on the same data to ground conclusions on their respective learning phases. We learn that across stages in the low and medium-low data regimes, small models benefit significantly from easier coding questions than from harder ones. We also find that, surprisingly, the correctness of outputs in training data makes no difference to distillation outcomes. Our work represents a step forward in understanding the training dynamics of code reasoning distillation outside intuition. We are open-sourcing dataset splits used for all our experiments at https://collinear.ai/valley-of-reasoning

1 Introduction

Test-time scaling techniques have enabled large language models (LLMs) to perform complex multi-step reasoning through chain-of-thought (CoT) style processes. Models such as DeepSeek-R1 Guo et al. [2025] and QwQ-32B Team [2025] demonstrated strong reasoning abilities, particularly in the domains of mathematics, science and coding. These, in turn, have amplified efforts to distill test-time compute gains via supervised fine-tuning (SFT) into smaller fine-tuned models. Recent work has demonstrated that LLMs can be taught to produce such long CoT reasoning through distillation in a surprisingly data-efficient manner Muennighoff et al. [2025], Ye et al. [2025]. For example, Li et al. [2025] fine-tuned a 32B model on only $17k$ reasoning samples and achieved dramatic gains on competitive coding benchmarks comparable to larger models. Despite the growing body of work on curating high-quality post-training data for reasoning distillation for frontier coding performance Guha et al. [2025], Ahmad et al. [2025b], only limited focus has been given to the training dynamics of these student models and how these dynamics influence the acquisition of reasoning abilities.

In this work, we investigate how reasoning abilities are distilled into non-reasoning student models, specifically asking whether their reasoning abilities emerge linearly. We hypothesize that, for small non-reasoning LLMs in low to medium-low data regime, distillation does not yield monotonic improvements: performance initially declines when trained on small data, then steadily improves as data scales, which we refer to as the *valley of reasoning*. Grounding our findings on model performances on LIVECODEBENCH (LCB) Jain et al. [2024], a competitive coding benchmark, we validate this claim and observe a sharper-than-log-linear trend once the model gets out of the valley (Fig 1).

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: .

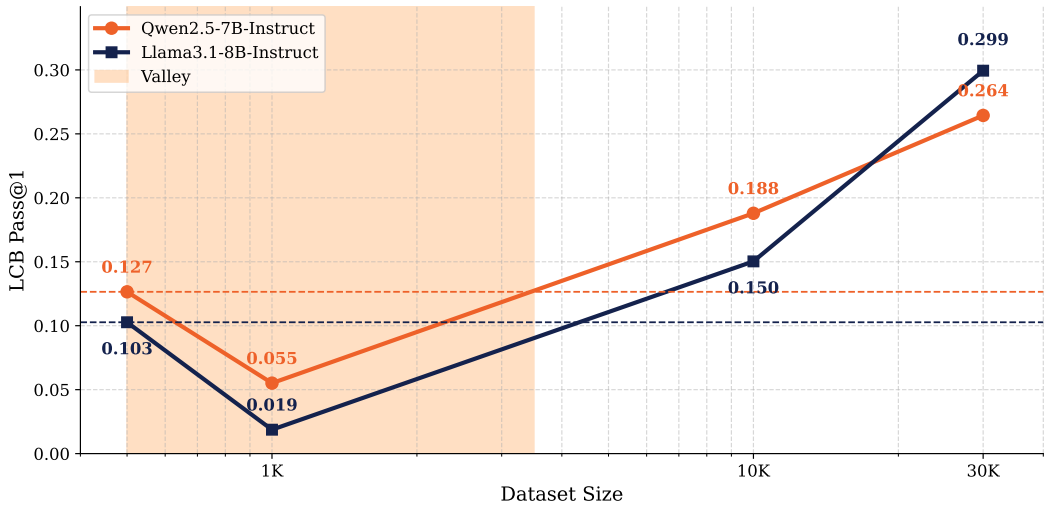

Figure 1: Evaluation scores on LCB of two distilled small models show that performances initially decreases by half but then steadily increases in a log-linear trend toward the 30K data upper bound.

Inspired by the findings of Li et al. [2025], which argues that output structure rather than code correctness in data drives improvement, we further whether this conclusion holds uniformly across stages of distillation. We find that both during the non-reasoning phase and the intermediate phase where the model acquires decent reasoning skills, code correctness in data has little impact on evaluation performances. However, for both stages, we consistently observe that distillation data containing easier coding questions outperform the one with harder questions. This suggests that at a low data regime, small models can better study after easier examples to yield immediate gains.

2 Related work

SFT for Reasoning Distillation SFT on reasoning traces generated by reasoning models has proved effective at improving smaller models Guo et al. [2025]. Motivated by this, many works have investigated methods for performing distillation. In particular, recent data-efficiency results show that small, carefully curated sets can already induce strong reasoning behavior in large (32B+) models Ye et al. [2025], Li et al. [2025], complementing early self-improvement/distillation antecedents Zelikman et al. [2022, 2024]. Several studies probe the components of reasoning traces: prefix-only supervision can enable highly efficient distillation Ji et al. [2025]; reasoning quality of the distillation traces can impact downstream performance Luo et al. [2025], Yu et al. [2025]; and the CoT lengths in the mixture can be tuned to optimize both token efficiency and model capabilities Wu et al. [2025], Ma et al. [2025], Xu et al. [2025]. Strikingly, Li et al. [2025], Ahmad et al. [2025b], and Guha et al. [2025] have found that for models in low-data regimes, the correctness of the reasoning trace does not impact distillation quality. However, these works do not investigate how data quality impacts different stages of training, a matter of practical importance in large-scale post-training processes.

Scaling Distillation for Coding OpenThoughts Guha et al. [2025], OpenCodeReasoning (OCR) Ahmad et al. [2025b], and rStar-Coder Liu et al. [2025] find that competitive coding performance can be improved by scaling the reasoning dataset to millions with high question duplication counts. These works evaluate on robust, contamination-controlled code benchmarks such as LCB Jain et al. [2024] and CodeElo Quan et al. [2025], reinforcing the observed SFT-only gains. Together, they demonstrate the potential of large-scale SFT for dramatic improvements in base-model capabilities, but they do not focus on the model performance during intermediate data quantities. This motivates us to conduct an in-depth study on the training dynamics and progress on downstream evaluations.

Size	Metric	Qwen 2.5	Llama 3.1
1K	Completion rate	0.1842	0.4649
	Think tag rate	0.1406	0.0777
10K	Completion rate	0.3509	0.3459
	Think tag rate	0.3496	0.3496
30K	Completion rate	0.5802	0.7080
	Think tag rate	0.5677	0.7068

Table 1: Completion and think tag rates of Qwen2.5 and Llama3.1 across training data sizes.

Dataset	Qwen 2.5	Qwen 2.5-30K
Baseline	0.126	0.264
Correct 6K	0.185	0.347
Incorrect 6K	0.182	0.350
Hard 4K	0.137	0.296
Easy 4K	0.179	0.352

Table 2: LCB scores of Qwen2.5 at two stages were further trained on four partitions of data. Easy data significantly outperforms hard data, but correctness has little impact.

3 The Valley of Reasoning

In regards to the training dynamics of reasoning distillation, we want to answer three research questions (RQs): (i) Does code reasoning performance scale monotonously with data quantity?, (ii) Does the correctness of the teacher's responses matter for scaling student performance? and (iii) Does the difficulty of the input problems have an impact on the student performance? We now discuss how these three questions inform our data curation strategies, and detail the findings in section 4.

3.1 Datasets

To answer RQ-1, we create three quantity-controlled datasets based on a single code reasoning dataset of 30,000 examples. The questions are sourced from OpenCodeReasoning2 (OCR2) Ahmad et al. [2025a], which contains 34,125 unique competitive coding problems compiled from 4 data sources. The answers are generated using two models, DeepSeek-R1-0528 and KAT-V1-40B Zhan et al. [2025], with an average duplication count of 7, chosen because of their over 70% accuracy on LCB. As a reasoning dataset, each example contains a response that has `<think></think>` tags wrapped around the teacher model's thinking traces.

From this base set, we create a random subset of 10K examples, and then another random subset of 1K examples out of the 10K subset. We explicitly conduct random sampling to ensure the subsets follow the same distribution as the supersets and differ only in quantity. We train a model of choice on the three datasets and report their LCB scores to study how dataset size impacts distillation outcomes.

To answer RQ-2, we take another in-distribution dataset of 13,583 reasoning examples whose questions are only sourced from the TACO Li et al. [2023] dataset. As TACO provides test cases for each question, we label each model response as either `correct` or `incorrect` based on whether it passes all test cases for the given question. We then select two random subsets of 6K examples, one consisting of only correct responses and the other only incorrect responses. The performance of models trained on these two separate datasets answers whether code correctness matters.

To answer RQ-3, we leverage the existing `difficulty` labels in TACO to partition examples into two groups: examples labeled `hard`, `very_hard`, or `medium_hard` belong to the group of hard questions and the ones labeled `easy` or `medium` belong to the group of easy questions. We randomly select from the two groups to create two mutually exclusive datasets of size 4K. Models on the two datasets can provide useful signals on whether the model benefits from learning to solve harder questions.

3.2 Training setup

We train on two small instruction-tuned models as the study of interest: Qwen2.5-7B-Instruct (Qwen2.5) and Llama3.1-8B-Instruct (Llama3.1) Dubey et al. [2024]. Neither has the ability to output CoT traces wrapped in `<think>` tags nor a dedicated special `<think>` token in the tokenizer.

For Qwen2.5, we conduct a total of eleven experiments, which consist of the three runs on the 1K, 10K, and 30K datasets to answer RQ-1, and four on the correctness- and difficulty-controlled datasets for both the instruct model and the 30K finetuned checkpoint (Qwen2.5-30K) to answer RQ-2 and RQ-3. Running the same four experiments on two checkpoints gives us signals on how the impact of the same data features changes as the model improves its reasoning capacity. For Llama3.1, we conduct three runs on the 1K, 10K, and 30K datasets to study the data scaling trend relevant to RQ-1.

We train each job using torchtune on $8\times$ Nvidia H100 GPUs. We uniformly use a global batch size of 128, a learning rate of $8e-5$ with a warmup ratio of 0.10, and the AdamW optimizer Loshchilov and Hutter [2017] to ensure fair comparisons. We use a max sequence length of 32,768 due to Qwen's architecture constraints. Each job takes a total of 5 epochs and is evaluated on the final checkpoint.

4 Empirical Findings

We now discuss the empirical findings that answer our three research questions in section 3.

RQ1: There is a "valley of code reasoning": the student model's distillation performance initially drops, then it increases as the training data quantity goes up. As is illustrated in Fig 1, for both Qwen2.5 and Llama3.1, the same scaling trend holds: The model's LCB score first decreases by more than half from the baseline when trained on 1K examples, then improves and surpasses the baseline by around 50% when the data size reaches 10K. On Qwen2.5 and Llama3.1, training on 30K examples further lifts the performance respectively by 50% and 100% upon the 10K checkpoint, exhibiting or even surpassing the anticipated log-linear trend of improvement.

We observe similar trends in two auxiliary metrics that correlate strongly with the quality of the model's reasoning outputs. One of them is the *completion rate*, defined as the percentage of all model responses that finish within 32K tokens. The completion rate is highly correlated with effective reasoning as incomplete responses often repeat the same phrases before the eventual cutoff. We observe a steady 1.65-1.9 log-linear increase in the completion rate from the 1K to the 30K checkpoint.

Another metric is the `<think>` tag occurrence rate, defined as the percentage of all responses that begin with a `<think>` tag. A higher `<think>` tag occurrence rate shows the model is effectively learning the structure of its outputs. Surprisingly, the `<think>` tag is hard for the model to learn, as both models starts with below 20%. As data size scales up, we observe another log linear trend of 1.6-2.4 increase in the `<think>` tag occurrence rate all the way to the later checkpoints.

RQ2: Training examples that have correct responses do not improve small models' performances. For both Qwen2.5 and the SFT checkpoint with 30K data (Qwen2.5-30K), we look at the performance lift of further training on 6K correct responses versus 6K incorrect responses (Table 2). Compared to each baseline, the two subsets both lift the LCB score by around 50%, which suggests that the correctness of responses has no effect on distillation outcomes at both stages of training.

Furthermore, as the model continues to improve on the additional 6K at a rate comparable to the initial rise to 30K, it hints that model performance is far from saturation at the current 30K data scale.

RQ3: Small models benefit more from easy and medium-level examples than from hard examples. For both Qwen2.5 and Qwen2.5-30K, we also compare their evaluation outcomes when trained on an additional 4K data with hard-coded questions versus 4K easy and medium ones (Table 2). For both models, training on exclusively hard questions provides only a modest boost: Qwen2.5 improves by 7%, whereas Qwen2.5-30K improves by 11%. In contrast, when training on only easy and medium-difficult data, Qwen2.5 improves by 41%, while Qwen2.5-30K improves by 33%. Therefore, we conclude that for small models in the medium-low data regime, examples containing easier questions provide superior benefits to downstream code reasoning performances.

Curiously, in all training runs that answer RQ-2 and RQ-3, we observe that the completion rate and the `<think>` tag occurrence rate are only weakly correlated with evaluation performances. However, they are strongly correlated with data quantity, which jointly predicts the quality of code reasoning. For example, for both Qwen2.5 and Qwen2.5-30K, training on hard questions yields the same completion rate and `<think>` tag occurrence rate as is in training on easy questions. This suggests that the benefits of easier problems go beyond mere structural imitations of the teacher's reasoning traces.

5 Conclusion

In this work, we have demonstrated that for small LLMs in a low to medium-low data regime of around 1K-30K training samples, distillation performance does not scale monotonously with data quantity. Instead, the model performance follows a highly predictable pattern in the shape of a valley: It initially decreases by half, then increases log-linearly with a higher completion rate and more `<think>` tag occurrences. Backed by this observation, we study the impact of several data features in different stages of model training. We show that for small LLMs across stages, reasoning examples with easier questions are more beneficial, but the correctness of model responses hardly matters. We will take it upon ourselves to further work on showing how the trend scales up to the medium-high and high data regimes above 100K, and if the same conclusions about correctness and difficulty hold in those regions.

References

W. U. Ahmad, S. Majumdar, A. Ficek, S. Narenthiran, M. Samadi, J. Huang, S. Jain, V. Noroozi, and B. Ginsburg. Opencodereasoning-ii: A simple test time scaling approach via self-critique. arXiv preprint arXiv:2507.09075, 2025a.

W. U. Ahmad, S. Narenthiran, S. Majumdar, A. Ficek, S. Jain, J. Huang, V. Noroozi, and B. Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding. 2025b. URL https://arxiv.org/abs/2504.01943.

A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. arXiv e-prints, pages arXiv–2407, 2024.

E. Guha, R. Marten, S. Keh, N. Raoof, G. Smyrnis, H. Bansal, M. Nezhurina, J. Mercat, T. Vu, Z. Sprague, et al. Openthoughts: Data recipes for reasoning models. arXiv preprint arXiv:2506.04178, 2025.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. 2024. URL https://arxiv.org/abs/2403.07974.

K. Ji, J. Xu, T. Liang, Q. Liu, Z. He, X. Chen, X. Liu, Z. Wang, J. Chen, Z. Wang, et al. The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. 2025. URL https://arxiv.org/abs/2503.02875.

D. Li, S. Cao, T. Griggs, S. Liu, X. Mo, E. Tang, S. Hegde, K. Hakhamaneshi, S. G. Patil, M. Zaharia, J. E. Gonzalez, and I. Stoica. LLMs Can Easily Learn to Reason from Demonstrations: Structure, not content, is what matters! 2025. URL https://arxiv.org/abs/2502.07374.

R. Li, J. Fu, B.-W. Zhang, T. Huang, Z. Sun, C. Lyu, G. Liu, Z. Jin, and G. Li. Taco: Topics in algorithmic code generation dataset. arXiv preprint arXiv:2312.14852, 2023.

Y. Liu, L. L. Zhang, Y. Zhu, B. Dong, X. Zhou, N. Shang, F. Yang, and M. Yang. rstar-coder: Scaling competitive code reasoning with a large-scale verified dataset. arXiv preprint arXiv:2505.21297, 2025.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.

Y. Luo et al. Deconstructing long chain-of-thought. 2025. URL https://arxiv.org/abs/2503.16385.

X. Ma, G. Wan, R. Yu, G. Fang, and X. Wang. Cot-valve: Length-compressible chain-of-thought tuning. In ACL, 2025. URL https://arxiv.org/abs/2502.09601.

N. Muennighoff, K. Singh, P. Moturi, A. Zhan, X. Li, Y. Liu, J. Han, L. Fei-Fei, H. Lee, and S. Fidler. s1: Simple Test-Time Scaling. In International Conference on Learning Representations (ICLR), 2025. URL https://iclr.cc/virtual/2025/32769.

S. Quan, J. Yang, B. Yu, B. Zheng, D. Liu, A. Yang, X. Ren, B. Gao, Y. Miao, Z. Wang, et al. CodeElo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. 2025. URL https://arxiv.org/abs/2501.01257.

Q. Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Y. Wu, Y. Wang, T. Du, S. Jegelka, and Y. Wang. When more is less: Understanding chain-of-thought length in llms. 2025. URL https://arxiv.org/abs/2502.07266.

S. Xu, W. Xie, L. Zhao, and P. He. Chain of draft: Thinking faster by writing less. 2025. URL https://arxiv.org/abs/2502.18600.

Y. Ye, Z. Huang, Y. Xiao, E. Chern, S. Xia, and P. Liu. Limo: Less is more for reasoning. In Proceedings of COLM, 2025. URL https://arxiv.org/abs/2502.03387.

B. Yu, H. Yuan, Y. Wei, B. Wang, W. Qi, and K. Chen. Long-short chain-of-thought mixture supervised fine-tuning: Eliciting efficient reasoning in large language models. 2025. URL https://arxiv.org/abs/2505.03469.

E. Zelikman, Y. Wu, M. Muqeeth, and N. D. Goodman. Star: Bootstrapping reasoning with reasoning. 2022. URL https://arxiv.org/abs/2203.14465.

E. Zelikman, V. Harik, M. Muqeeth, and N. D. Goodman. Quiet-star: Language models can teach themselves to think before speaking. 2024. URL https://arxiv.org/abs/2403.09629.

Z. Zhan, K. Deng, H. Tang, W. Xiang, K. Wu, W. Li, W. Zhu, J. Xu, L. Huang, Z. Feng, et al. Kat-v1: Kwai-autothink technical report. arXiv preprint arXiv:2507.08297, 2025.