DON'T REPEAT YOURSELF: SELF-SUPERVISED FINE-TUNING FOR COVERAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

In verifiable domains, such as math and coding, where multiple solutions can be generated in parallel and scored by a verifier, getting just one correct solution among many attempts can be more important than the individual pass rate of each attempt, especially for difficult tasks where a correct solution might be a needle-in-the-haystack. Post-training methods such as supervised fine-tuning (SFT) and reinforcement learning based preference fine-tuning in Large Language Models (LLMs) have been shown to concentrate their outputs around a few modes. A common method to increase LLM output diversity is to increase a "temperature" parameter, but the effectiveness of this has been shown to be limited (Banayeeanzade et al., 2026; Holtzman et al., 2020). To address this issue we introduce Don't Repeat Yourself Supervised Fine-Tuning (DRY-SFT): a post-training method that increases output diversity and improves coverage — coverage being how likely it is for there to be at least one correct solution among many generated attempts. DRY-SFT has two simple stages: first, for each problem, sequentially generate K solutions, where, for each subsequent generation, the prompt contains the problem plus all of the prior problem attempts and asks for a different solution. Second, fine-tune the LLM on each solution attempt as if it was generated independently by stripping away the prior attempts from the context. This process involves no reward, verifier, or correctness filter. The result is a model that is trained to produce many diverse solutions. We evaluate DRY-SFT on three coding benchmarks and show that it raises pass@100+ by 10.8, 12.5, and 12.4 points across HumanEval+, MBPP+, and DS-1000 at a small cost to pass@1+. We also measure the structural diversity using abstract syntax tree (AST) edit distance of passing solutions and find that this rises significantly across all three benchmarks. Additionally, DRY-SFT solves 244 of 600 problems the base model was not capable of solving in the same 200 attempts. Finally, across nine open-weight models, lower structural diversity of the base model significantly predicts higher DRY-SFT performance gains, indicating that our method is especially effective on more mode-collapsed LLMs.

1 INTRODUCTION

In verifiable domains, such as math and coding, where multiple solutions can be generated in parallel and scored by a verifier, getting just one correct solution among many attempts can be more important than the individual pass rate of each attempt, especially for difficult tasks where a correct solution might be a needle-in-the-haystack (Li et al., 2022; Snell et al., 2024). In this setting, a model's coverage, which is the probability that at least one of its k samples is correct (pass@k), determines its performance. Importantly, to improve coverage, a model must generate attempts that differ meaningfully from other attempts in the context of the task objective. In other words, pass@k only rises if later samples solve what earlier ones couldn't (Brown et al., 2024). A model that produces only a few distinct solution modes for a given problem does not benefit from large amounts of repeated sampling, resulting in pass@k plateauing at low k. Many post-training algorithms result in this exact situation, often referred to as mode-collapse. Kirk et al. (2024) show that RLHF measurably sharpens and collapses the output distribution, and Zhang et al. (2026b) attribute this collapse to a typicality bias in preference data. Yue et al. (2025) further show that reinforcement learning with

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.

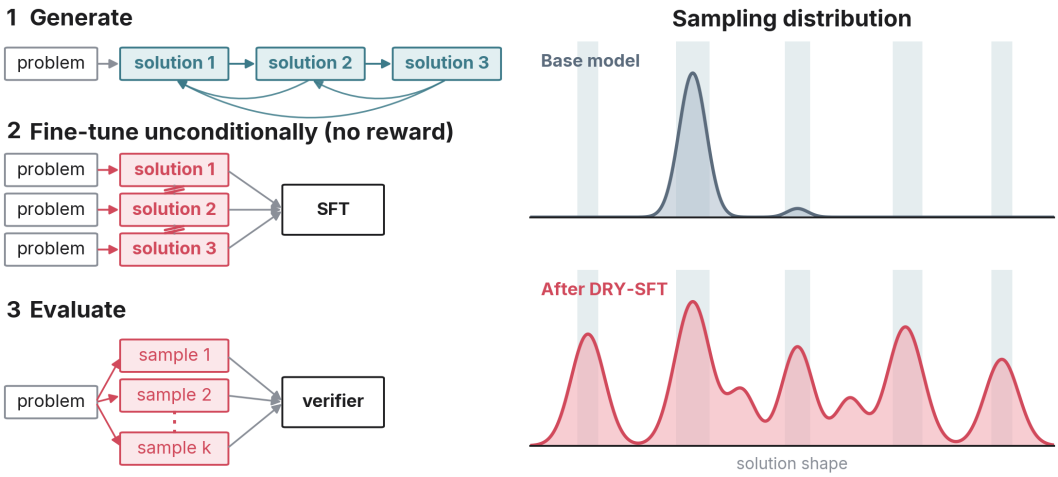


Figure 1: DRY-SFT. **Generate:** a chain of solutions per problem, each conditioned on those before it. **Fine-tune:** the conditioning is discarded and each solution is paired with the problem for SFT: no reward, no verifier, no filter. **Evaluate:** solutions are drawn independently and in parallel from the prompt. Right: the illustrative change in the distribution over solution types. The distributions illustrate the mechanism and are not measured. The measured structural change is in Figure 2.

verifiable rewards largely moves pass@k performance into pass@1 without moving high pass@k. Thus, most post-training methods optimize for pass@1 at the cost of pass@k.

A common method for increasing output variance is to increase a "temperature" parameter. However, this has been shown to lack necessary control over the output distribution. Banayeeanzade et al. (2026) demonstrate the limitation of temperature and attribute this limitation to the characteristics of the token-level output distribution. They find that the probability mass of tokens at a given token generation step concentrates on a few valid tokens above a heavy tail that mixes valid tokens with invalid ones. This means that a temperature parameter high enough to reach the rest of the valid set also includes the invalid tokens, and a temperature low enough to stay within valid tokens limits diversity. No ranking or probability-based token decoding method can avoid this token validity and output diversity tradeoff. Decoding methods can only re-weight the distribution they are given (Vijayakumar et al., 2018; Holtzman et al., 2020). Therefore, to increase the output variance without significantly degrading outputs you must change the output distribution itself instead of simply reweighting it.

To address this, we introduce Don't Repeat Yourself Supervised Fine-Tuning (DRY-SFT, Figure 1): a post-training method that changes the output distribution by fine-tuning on a diverse set of self-generated rollouts which improves coverage and output diversity. DRY-SFT has two simple stages: first, for each problem, sequentially generate K solutions, where, for each subsequent generation, the prompt contains the problem plus all of the prior problem attempts and asks for a different solution. Second, fine-tune the model on each solution attempt as if it were generated independently by removing the prior attempts from the context. By including the prior solutions in the context, the model produces a naturally diverse set of solutions. This diversity is trained into the model weights such that, at inference time, parallel independent sampling produces samples with more diversity. On three coding benchmarks DRY-SFT raises pass@100+ by 10.8, 12.5, and 12.4 points over the base model. A matched i.i.d. control that matches the training protocol of DRY-SFT but generates the training solutions independently moves pass@100+ by only 0.7, 1.8, and -3.4 points. Additionally, P-SFT (Fan et al., 2026), the closest published method, applied to the same self-generated solutions, moves pass@100+ by only 1.0, 2.5, and -2.4 points. Our method offers a way to significantly widen a mode-collapsed model's output distribution without a reward, a verifier, or any correctness proxy. To summarize, we make the following contributions:

1. **A reward-free method for increasing output diversity.** DRY-SFT is a post-training method for increasing the diversity of a model's output distribution without any reward, verifier, or filtering.

2. **A characterization of the DRY-SFT objective.** DRY-SFT learns a mixture over K distinctly conditioned contexts for each problem. We contrast against a control that isolates this mechanism.

3. **Coverage and diversity gains across three code benchmarks.** DRY-SFT significantly raises pass@100+, increases structural diversity, generalizes gains out-of-sample and across datasets, and solves problems unsolved by the base model across HumanEval+, MBPP+, and DS-1000. Neither the control, P-SFT (Fan et al., 2026), nor a temperature sweep reproduces the same gains.

4. **A test to predict the effectiveness of DRY-SFT.** Across nine open-weight models, the structural diversity of the base model correlates with the effectiveness of DRY-SFT (R^2=0.65, p=0.0088).

2 Related Work

Rewarding coverage with a verifier. The simple way to increase coverage is to target it directly. Chen et al. (2025) make pass@k itself the reward by rewarding a group of k rollouts when at least one is correct. Cheng et al. (2025) instead add a token-level entropy term to the advantage, encouraging exploration at the pivotal tokens that determine or connect logical steps. However, both of these methods require a verifier, whereas DRY-SFT does not.

Filtering self-generated data. A different set of methods applies a correctness filter and then fine-tunes on the correct attempts instead of maximizing a reward. STaR (Zelikman et al., 2022) and ReST^{EM} (Singh et al., 2024) both generate solutions for each problem, filter for correctness, refit the base model on the correct set with SFT, and repeat. GiFT (Li et al., 2025) alternates between generating code from a description and generating a description of that generated code, Gibbs-style, producing a diverse set of descriptions and solutions for each problem which is then filtered for correctness and trained into the model. Here, the generations are conditioned on rephrasings of the problem, where ours is conditioned on previous solutions. Recursive training on synthetic data can truncate the tails of a model's output distribution (Shumailov et al., 2024), but its effects depend on whether synthetic samples replace earlier data or accumulate alongside the original data (Gerstgrasser et al., 2024). These results concern recursive data reuse, not correctness filtering alone. Our single-cycle experiments do not establish stability under repeated self-training.

Generation under privileged context. Some methods generate solutions under a context that is removed during training. Snell et al. (2022) and Yu et al. (2024) both prompt a model to generate a reasoning trace and an answer to a given problem and then fine-tune the model to predict its answer from the prompt alone. However, neither of these methods elicit diversity. SESA does elicit diversity (Kang & Zhang, 2025) by generating solution sketches sequentially such that each is conditioned on those before it. These sketches are then added to the prompt separately during reinforcement learning rollouts. Importantly, SESA requires a sequential sketch generation step before any rollouts can be generated. Our method instead amortizes this sequential generation step by training it into the model with SFT without rewards.

Changing the objective, or presupposing variation in the data. Other methods aim to preserve diversity directly through fine-tuning. P-SFT (Fan et al., 2026) trains on each existing solution behind a randomly generated number prompt prefix, resampled every iteration so the prefix separates training contexts rather than acting as an index. The same prefixes also work at inference alone, without any training. SSFT (Jia et al., 2026) assigns each existing solution for a problem a reserved token and then fine-tunes on the pairings that the model is the least surprised about (lowest log-loss). SED-SFT (Chen et al., 2026) adds entropy regularization to the SFT objective. Klypa & Cherednichenko (2026) use a Tempered Focal loss to target the neglect of low-frequency patterns and the forgetting of pretrained knowledge that they identify as the two drivers of diversity loss. DivPO (Lanchantin et al., 2025) builds pairs by selecting chosen responses that are rare but high quality against rejected ones that are common but low quality. Each of these methods either changes the loss or requires a set of multiple solutions per problem that it does not itself produce, whereas DRY-SFT requires neither.

Injecting variation during inference. A fourth set of methods varies the context at generation time. Wang et al. (2025) study prompt perturbation for Best-of-N scaling, deriving a diversity–fidelity trade-off that governs how much perturbation helps. Zhang et al. (2026a) propose generating diverse intermediate specifications to condition on. Verbalized Sampling (Zhang et al., 2026b) asks the model to verbalize a probability distribution over responses. None of these require training, but because of that, they require applying the prompting method to every generation at inference time rather than training it into the model.

How DRY-SFT is different. DRY-SFT is a post-training method that does not require changing the loss, optimizer, or objective and changes only how the training set is constructed. Neither a reward, a verifier, a correctness filter, nor a pre-existing multi-response corpus is required. Diverse solutions are elicited through a natural sequential generation method. This diversity is trained into the model once and results in a model where independent sampling produces meaningfully diverse solutions from a bare prompt.

3 Don't Repeat Yourself Supervised Fine-Tuning

Generation and fine-tuning. DRY-SFT has two stages (Figure 1). In Stage 1, for each problem, K solutions are generated sequentially such that each subsequent generation is prompted with the problem plus the previous generations (exact prompts are in Appendix G). In Stage 2, we remove the prior solutions and diversity prompt from the context. With a dataset $\mathcal{D}$ of N problems, this produces $N \cdot K$ training pairs used to fine-tune the model with LoRA SFT. Neither stage uses a reward, verifier, or any kind of correctness filter, meaning that every generated solution is trained on. At test-time the model is sampled independently and in parallel using a standard prompt.

Objective. Let $\pi_\theta(y \mid x)$ be a model's distribution over solutions y to a problem x. Stage 1 draws K solutions sequentially,

$$y_1 \sim \pi_\theta(\cdot \mid x), \qquad y_i \sim \pi_\theta(\cdot \mid x, y_{<i}), \quad i = 2, \ldots, K, \tag{1}$$

where $y_{<i}$ contains all solution attempts on problem x before generation i. Stage 2 then trains on each solution y_i conditioned only on the problem x,

$$\max_\theta \sum_{x \in \mathcal{D}} \sum_{i=1}^{K} \log \pi_\theta(y_i \mid x). \tag{2}$$

This fits a mixture over the K conditionally generated solutions from Equation 1. A matched i.i.d. control doesn't condition on $y_{<i}$, so every position is drawn from $\pi_\theta(\cdot \mid x)$ and Equation 2 fits the same distribution the model already has. This illustrates how DRY-SFT differs from plain self-generated data fine-tuning. The key step is that self-generated data is conditioned on previous attempts in Equation 1. Nguyen et al. (2026) empirically motivate this. They show that when each problem has several reasoning modes, coverage is high. However, when each problem is fit to a single mode, coverage degrades. This applies even when the overall dataset is balanced across modes.

4 Experiments

Data. We evaluate DRY-SFT on three coding datasets: HumanEval+ and MBPP+ from EvalPlus (Liu et al., 2023) with the extended test suites, and DS-1000 (1,000 data-science problems) with its official execution harness (Lai et al., 2023).

Coverage. Our primary metric of interest is pass@k which measures a model's coverage. It is estimated as the average probability that at least one of k independently generated samples is correct for any given problem in a dataset (Chen et al., 2021). We report pass@k+, where the plus indicates that a solution is only correct when it passes the extended EvalPlus test suite. On DS-1000 it indicates correctness on the official harness. We sample n=200 completions per problem at temperature 0.8 and nucleus p=0.95 (Holtzman et al., 2020), and report uncertainty with a 95% percentile bootstrap over problems with 2,000 resamples unless noted otherwise.

Structural diversity. Coverage can only measure whether a model solves a problem within a given number of attempts, but not whether the passing solutions differ from each other in meaningful ways. To address this, we measure structural diversity in the context of code: the mean pairwise Zhang–Shasha tree edit distance (Zhang & Shasha, 1989) between the abstract syntax trees (AST) of a problem's passing solutions, normalized by each problem's largest tree size to $[0, 1]$. Comparing syntax trees lets us differentiate surface-level changes, such as different variable names, from real differences in what a program does or how it works. We measure it on the first 10 samples per problem, over problems with at least two passing solutions.

Models and baselines. We compare DRY-SFT against three baselines on Qwen3-4B-Instruct-2507 (Qwen Team, 2025): the base model, a diversity-prompted i.i.d. SFT control, and P-SFT (Fan et al., 2026). The three fine-tuned arms use $K{=}5$ unfiltered self-generated solutions per problem (820 for HumanEval+, $1{,}890$ for MBPP+, $5{,}000$ for DS-1000), the same SFT objective and LoRA optimizer, and the same validation-based checkpoint selection. P-SFT uses exactly the control generations but prepends a random state: {} prefix to every training pair. We follow P-SFT's unstarred protocol, which removes prefixes and uses ordinary sampling during inference. To choose K, we performed a hyperparameter sweep through $K{=}10$ and chose $K{=}5$ since the coverage flattened between $K \in [3, 7]$ (Appendix C; Appendix D reports how target quality varies along the chain). For SFT, we choose the checkpoint with the lowest cross-entropy on a 10% problem-level holdout independently for each arm (Appendix B). Appendix A lists all hyperparameters in more detail.

4.1 Pass@k Performance

Coverage increases consistently across benchmarks. The advantage of DRY-SFT is shown in Figure 2. DRY-SFT improves pass@100+ over the base model by $+10.8$ points on HumanEval+ (from 0.851 to 0.960), by $+12.5$ points on MBPP+ (0.749 to 0.874), and by $+12.4$ points on DS-1000 (0.507 to 0.632) with non-overlapping bootstrap confidence intervals. The i.i.d. control moves pass@100+ by $+0.7$, $+1.8$, and -3.4 points. P-SFT moves pass@100+ by $+1.0$, $+2.5$, and -2.4 points. At $k = 100$, neither baseline produces the same improvement as DRY-SFT on any of the three benchmarks with non-overlapping bootstrapped 95% confidence intervals. Randomized prefixes therefore do not substitute for conditioning on prior solutions. It is not entirely obvious whether DRY-SFT trained at $K = 5$ should show improvement beyond pass@5+. However, these results show that pass@k+ continues to improve well beyond pass@5+. Additionally, Appendix E reports a DRY-SFT coverage ablation that compares correctness filtering versus no filtering before SFT and finds that the performance differences are relatively small.

The pass@1+ trade-off. Pass@1+ falls from 0.757 to 0.735 on HumanEval+, from 0.665 to 0.648 on MBPP+, and from 0.357 to 0.293 on DS-1000. However, the coverage curves cross by $k{=}2$ on HumanEval+ and MBPP+ benchmarks and by $k{=}3$ on DS-1000, so DRY-SFT beats the base model after a small number of samples.

Solves many unsolved problems. On HumanEval+, MBPP+, and DS-1000, DRY-SFT uniquely solves 18, 54, and 172 problems respectively that the base model was never able to solve within the same 200 attempts, 244 of the 600 problems the base model never solves. The base model uniquely solves 0, 3, and 34 problems respectively, however, DRY-SFT solves roughly 6.6 times more unique problems across the three datasets. This is a significant number of unsolved problems solved by DRY-SFT without requiring any correctness or reward signal.

4.2 Meaningful Diversity

Program structure increases with coverage. Figure 2 shows that the normalized AST edit distance among passing solutions rises from 0.072 to 0.264 on HumanEval+, from 0.051 to 0.286 on MBPP+, and from 0.057 to 0.308 on DS-1000. The matched i.i.d. control reaches 0.043, 0.058, and 0.073, while P-SFT reaches 0.053, 0.060, and 0.065. This supports our claim that the increase in output diversity is structural as opposed to surface level.

Diversity is elicited through sequential generation. The sequentially conditioned generations that we're trained on reach 0.403, 0.453, and 0.325 normalized AST edit distance, significantly above the fine-tuned model on each dataset. This shows that DRY-SFT takes the variation the base model could

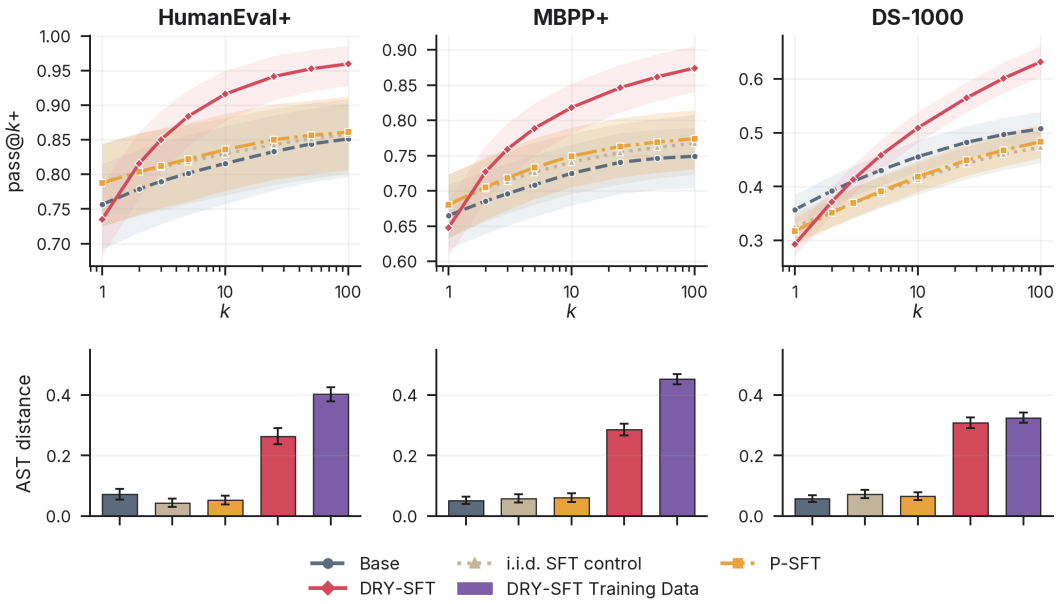


Figure 2: DRY-SFT increases coverage and structural diversity. **Top:** pass@k+ from n=200 independent samples per problem at T=0.8 and p=0.95. The i.i.d. SFT control and P-SFT use the same 5 independently self-generated attempts; P-SFT adds resampled random-state prefixes during training but not at inference. DRY-SFT uses 5 sequentially generated targets conditioned on prior solutions. Bands are pointwise 95% problem bootstrap. **Bottom:** structural diversity, the mean pairwise tree edit distance between the abstract syntax trees (ASTs) of a problem's passing solutions, normalized to $[0, 1]$ (Zhang & Shasha, 1989), over the first 10 samples (Section 4, Structural diversity). Bars have 95% bootstrapped confidence intervals over problems with at least two passing solutions. DRY-SFT Training Data is the set of Stage 1 solutions DRY-SFT was fine-tuned on, not a model arm.

already produce, but only under a specific context, and trains it into the model's default sampling distribution.

4.3 Temperature

Increasing temperature recovers some coverage but not structural diversity. In Figure 3, we evaluate whether raising the base model's sampling temperature reproduces the coverage and diversity gains of DRY-SFT. The best temperature on the grid reaches pass@100+ $= 0.910$ at T=2.6 on HumanEval+. DRY-SFT outperforms this by 5.0 points. However, the intervals overlap, so we read this as a point-estimate difference rather than a statistically significant difference. When comparing the structural diversity at that peak temperature, the normalized AST edit distance is 0.175, against 0.264 for DRY-SFT. DRY-SFT clearly has a more structurally diverse output distribution. Also, notice that there is a steep dropoff near T=3.0 for the base model. This highlights the issue that Banayeeanzade et al. (2026) identify when temperature is too high: the sampling distribution starts to put too much mass on invalid tokens, and the model no longer reliably produces valid generations. DRY-SFT avoids this failure mode and significantly increases both output diversity and coverage.

4.4 Held-out and Cross-dataset Performance

Performance transfers on a held-out set. The experiments in Sections 4.1 and 4.2 (Figure 2) generate solutions to and evaluate on the same problems. A dataset split is not an important concern in this setting since there is no correctness signal used for training. However, we address any leakage concerns in this experiment by training on one half and evaluating on the other for each of the three datasets. Figure 4 shows pass@100+ on the held-out set increases from 0.888 to 0.956 on HumanEval+, from 0.751 to 0.881 on MBPP+, and from 0.523 to 0.696 on DS-1000, ruling out leakage or problem memorization.

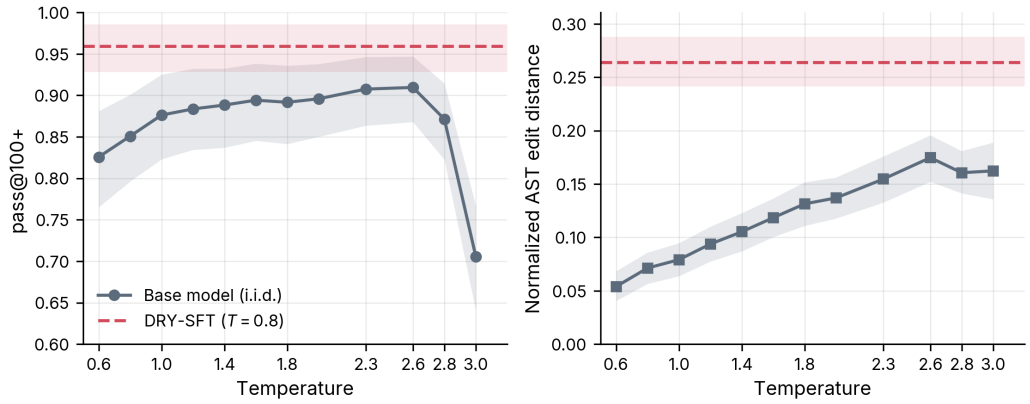


Figure 3: Temperature recovers coverage but not structure. Base-model pass@100+ (left) and structural diversity (normalized AST edit distance among passing solutions, defined in Section 4) (right) against temperature on HumanEval+, over a grid of 12 temperatures from 0.6 to 3.0. Coverage climbs to a peak at T=2.6 and falls away beyond it, while structural diversity reaches DRY-SFT's level at no temperature. Shaded regions are pointwise 95% problem-bootstrap confidence bands; dashed lines and bands mark DRY-SFT at T=0.8.

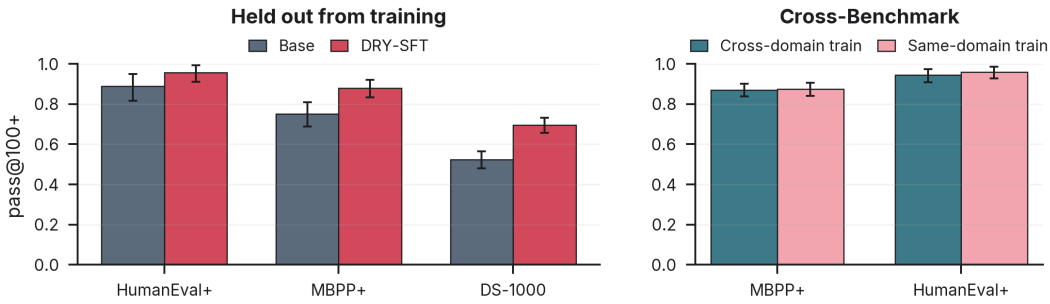


Figure 4: The gain is not tied to the problems the targets came from. **Left:** pass@100+ on the held-out half of a 50/50 split, with DRY-SFT trained only on the other half. **Right:** DRY-SFT trained on the opposite EvalPlus benchmark against same-domain training, evaluated on the benchmark named. Error bars are 95% problem bootstrap.

Performance transfers across datasets. We further investigate how well DRY-SFT generalizes coverage across datasets. Figure 4 shows DRY-SFT trained on HumanEval+ and evaluated on MBPP+ reaches pass@100+ $= 0.871$, only a -0.003 difference compared to the DRY-SFT model trained on MBPP+. Similarly, the reverse training and evaluation setup reveals only a -0.015 difference compared to the in-domain HumanEval+ trained model. This demonstrates that the increased coverage and diversity of DRY-SFT can generalize across datasets.

4.5 When DRY-SFT works

Base model diversity predicts method gains. We applied DRY-SFT to nine different open-weight models (Qwen Team, 2025; LG AI Research, 2024; Qwen Team, 2024; Granite Team, IBM, 2025; Team OLMo et al., 2024; Mistral AI, 2024; Guo et al., 2024) and plotted each model's change in pass@100+ against the normalized AST edit distance of its base model (Figure 5). The base structural diversity explains 65% of the variation in Δ pass@100+ points across the models (R^2=0.65, p=0.0088). The six models with the lowest base structural diversity all gain between 2.8 and 21.5 Δ pass@100+ points. This indicates that the more mode collapsed a model is, the more it will benefit from DRY-SFT. This also motivates a practical test to run before applying DRY-SFT to estimate the coverage gains.

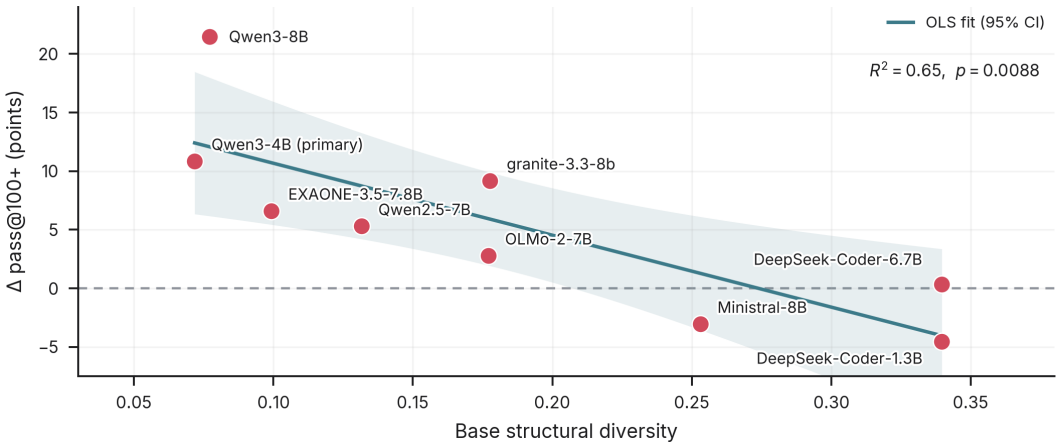


Figure 5: The gain tracks how concentrated the base model already is. Each point is one fine-tuned model: its base structural diversity — the normalized AST edit distance among its passing solutions — against the change in pass@100+ after DRY-SFT. The line is an ordinary least-squares fit over nine runs with a 95% confidence band; R^2 and p are printed in the panel. The fitted slope is -6.14 points per 0.1 of base diversity, with 95% CI $[-10.18, -2.10]$. Models below the dashed line lose coverage. Every model is measured on HumanEval+.

5 Conclusion

We introduced a new post-training method that improves coverage and introduces diversity into mode-collapsed LLMs. We have shown that this method significantly improves pass@100+ on three coding benchmarks while increasing meaningful, structural diversity of the output distribution. We have demonstrated that increasing temperature does not fully recover these gains and that continuing to raise temperature eventually degrades pass@100+. We have also established that DRY-SFT transfers performance and diversity across benchmarks. Lastly, we identified a significant relationship between base model diversity and the effectiveness of DRY-SFT, providing a practical LLM diversity test to estimate the coverage gains of DRY-SFT.

Limitations. DRY-SFT causes a decrease in pass@1+ across all datasets. However, the primary motivation of our method is to raise pass@k. Second, DRY-SFT did not improve coverage for models whose baseline output distribution is already relatively diverse. However, we position DRY-SFT as a remedy for mode-collapsed models which are by definition under-diverse. Third, Stage 1 generation is sequential: the i-th solution cannot begin until the first $i-1$ are complete. This also means that the context grows with each addition, so eliciting K training targets costs more than drawing K independent samples. However, this happens only once during data generation. There is no cost at inference. We also did not evaluate on any domains outside of coding. Future work should target this limitation. Lastly, our only external baseline is P-SFT. This is because we do not compare against reward-based coverage methods which require a verifier that DRY-SFT does not.

Future directions. A natural question is whether the elicited variation is present in the weights of the base model or whether the base model computes the variation over the course of sequential generation. We leave this direction open for future work. Answering it could explain both why DRY-SFT works and how its mechanism could be optimized for further improvements.

DRY-SFT significantly raises pass@k, and it is well documented that reinforcement learning moves pass@k to pass@1 (Yue et al., 2025). Hence, reinforcement learning could serve as a natural complement to DRY-SFT, either after DRY-SFT or alternating between the two.

A third direction could explore controlling the type of output diversity. Temperature is the traditional scalar used for controlling variation, but by tuning the sequential rollout prompt of DRY-SFT to differentiate later solutions along an axis of interest, such as lexically versus semantically for example, the model might learn to spread its output distribution along that axis.

AI USE STATEMENT

In this work, we used generative AI tools to support data analysis, implement methods, and polish writing. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A lists every training and sampling hyperparameter, and Appendix G gives the verbatim system and user prompt templates for both stages. Section 4 states the benchmarks, the number of samples per problem and the metrics.

REFERENCES

Amin Banayeeanzade, Qingchuan Yang, Dhruv Tarsadiya, Fatemeh Bahrani, Leonardo Blas, Alfy Samuel, Robin Jia, Meisam Razaviyayn, and Sai Praneeth Karimireddy. Sampling more, getting less: Calibration is the diversity bottleneck in LLMs. *arXiv preprint arXiv:2605.11128*, 2026.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

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, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yijie Chen, Yijin Liu, and Fandong Meng. SED-SFT: Selectively encouraging diversity in supervised fine-tuning. *arXiv preprint arXiv:2602.07464*, 2026.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.

Zhiyuan Fan, Guanqiao Chen, Yanyi Huang, Mingkuan Zhao, Dadi Guo, and Yi R. Fung. Learning diverse responses with prefix-conditioned supervised fine-tuning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 247–276, 2026.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

Granite Team, IBM. Granite-3.3-8B-Instruct. `https://huggingface.co/ibm-granite/granite-3.3-8b-instruct`, 2025.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming — the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Sheng Jia, Xiao Wang, and Shiva Prasad Kasiviswanathan. Training large language models to reason in parallel with global forking tokens. In *International Conference on Learning Representations*, 2026. Preprint: arXiv:2510.05132.

Shijia Kang and Muhan Zhang. The road less traveled: Enhancing exploration in LLMs via sequential sampling. *arXiv preprint arXiv:2510.15502*, 2025.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *International Conference on Learning Representations*, 2024. Preprint: arXiv:2310.06452.

Roman Klypa and Oleksandr Cherednichenko. Diversity in large language models under supervised fine-tuning. *arXiv preprint arXiv:2605.00195*, 2026.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pp. 18319–18345, 2023.

Jack Lanchantin, Angelica Chen, Shehzaad Dhuliawala, Ping Yu, Jason Weston, Sainbayar Sukhbaatar, and Ilia Kulikov. Diverse preference optimization. *arXiv preprint arXiv:2501.18101*, 2025.

LG AI Research. EXAONE 3.5: Series of large language models for real-world use cases. *arXiv preprint arXiv:2412.04862*, 2024.

Haochen Li, Wanjin Feng, Xin Zhou, and Zhiqi Shen. GiFT: Gibbs fine-tuning for code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

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*, 2023.

Mistral AI. Un ministral, des ministraux. `https://mistral.ai/news/ministraux/`, 2024.

Ngoc-Hieu Nguyen, Parshin Shojaee, Phuc Minh Nguyen, Nan Zhang, Chandan K. Reddy, Khoa D. Doan, and Rui Zhang. Why do reasoning models lose coverage? the role of data and forks in the road. *arXiv preprint arXiv:2605.17026*, 2026.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman

Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656v1*, 2024.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. In *AAAI Conference on Artificial Intelligence*, 2018. Preprint: arXiv:1610.02424.

Tianchun Wang, Zichuan Liu, Yuanzhou Chen, Jonathan Light, Weiyang Liu, Haifeng Chen, Xiang Zhang, and Wei Cheng. On the effect of sampling diversity in scaling LLM inference. *arXiv preprint arXiv:2502.11027*, 2025.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Cheng Zhang, Rui Xin, and Chudi Zhong. Where you inject diversity matters: A unified framework for diverse generation. *arXiv preprint arXiv:2606.10302*, 2026a.

Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock LLM diversity. In *International Conference on Machine Learning*, 2026b.

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

A Hyperparameters

The generation and fine-tuning settings are shared by DRY-SFT, P-SFT, and the matched i.i.d. SFT control. P-SFT and the control use identical independent targets and differ only in randomized-prefix conditioning during training; DRY-SFT changes how those targets are generated. Table A.1 lists every setting used by the reported runs.

B Choosing the training epoch

For each fine-tuned arm, we hold out 10% of problems and train for five epochs, restoring the checkpoint with the lowest held-out cross-entropy. We split by problem rather than example because the K targets of a problem share a prompt, and an example-level split would put near-duplicates on both sides of the split.

A fixed three-epoch budget trains too long. To confirm that validation loss is not merely tracking likelihood, we score intermediate checkpoints directly. On HumanEval+ at $K{=}5$, pass@100+ is 0.957, 0.963, and 0.955 at epochs 1, 2, and 3, respectively. On MBPP+, pass@100+ is 0.884, 0.884, and 0.872 at epochs 1, 2, and 3, respectively. The fixed epoch-3 checkpoint is the worst of the three on both benchmarks. On MBPP+, the advantage of epoch 2 over epoch 3 is $+0.012$, with a paired problem-bootstrap interval of $[+0.001, +0.024]$. We also find that the validation loss reaches its minimum before the third epoch on every benchmark and rises monotonically thereafter. For example, on HumanEval+ the validation loss is 0.409, 0.389, 0.444, and 0.501 at epochs 1, 2, 3, and 5, respectively.

The selection is robust to the holdout size. Repeating the procedure with a 30% holdout (49 validation problems on HumanEval+ instead of 16) selects the same epoch on every arm. The epoch that minimizes the validation loss is also among the epochs that maximize pass@100+ of the epochs

Table A.1: Every hyperparameter of the reported runs. Appendix B covers how the epoch is selected and Appendix C the choice of K.

	Setting	Value
Generation	model	Qwen3-4B-Instruct-2507
	solutions per problem, K	5
	temperature / nucleus p	0.8 / 0.95
	correctness filtering	none
Fine-tuning	method	LoRA
	LoRA rank / scaling α	64 / 128
	LoRA dropout	0.05
	LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
	optimizer	PyTorch AdamW
	AdamW β_1 / β_2 / ϵ	0.9 / 0.999 / 10^{-8}
	weight decay	0
	learning rate	2×10^{-5}
	learning-rate schedule	cosine decay with 10% warmup
	maximum gradient norm	1.0
	batch size	16
	maximum sequence length	8,192 tokens
	epoch budget	5
	checkpoint selection	lowest cross-entropy on a 10% problem-level holdout
	selected epoch	2 (HumanEval+), 1 (MBPP+), 2 (DS-1000)
P-SFT	prefix template	`random state: {00000-99999}` followed by a space and two newlines
	training prefix draw	without replacement; resampled every presentation
	selected epoch	2 (HumanEval+), 1 (MBPP+), 2 (DS-1000)
	plotted inference	no prefix; temperature 0.8, nucleus p=0.95
Evaluation	samples per problem, n	200
	temperature / nucleus p	0.8 / 0.95
	pass@k+ estimator	unbiased (Chen et al., 2021)
	uncertainty	95% percentile bootstrap over problems, 2,000 resamples
Training examples	HumanEval+	820
	MBPP+	1,890
	DS-1000	5,000

we scored. This matters because the goal is to spread across solution structures, which cross-entropy does not measure.

What this does not settle. First, the number of epochs is not fully separable from K, since the number of updates scales with the size of the training set (Appendix C). Second, the selected epoch is not a property of the benchmark alone: the held-out arms select later epochs than their main counterparts (3 against 2 on HumanEval+, and 2 against 1 on MBPP+), as expected when training on half as many problems.

C Choosing the chain depth K

Shorter chains are not worse, and are usually better. We sweep the chain depth K, holding every other choice fixed. Figure C.1 compares depths against K=10 on the same problem resample. Every depth raises pass@100+ over the base model by 7.9 to 12.3 points, and paired problem-bootstrap intervals exclude zero on all three benchmarks. 11 of the 12 contrasts favor the shorter chain, and

four exclude zero: $K{=}3$, $K{=}5$, and $K{=}7$ on DS-1000 at $+3.1$ $[+1.1, +5.1]$, $+2.9$ $[+1.2, +4.6]$, and $+1.6$ $[+0.3, +3.0]$, and $K{=}5$ on MBPP+ at $+2.4$ $[+0.8, +4.2]$ points. No contrast favors $K{=}10$ by an interval excluding zero. Smaller K retains fewer positions and targets, but each arm is a prefix of the same chain, so $K{=}10$ is the deepest arm rather than the reported one.

The single-sample cost grows with depth. pass@1+ declines overall as K grows on all three benchmarks, although the point estimates are not strictly monotone. On the two EvalPlus benchmarks, the regression is not resolvable from zero at $K \leq 3$, and the coverage gain at short depths is close to free. On DS-1000, the regression is significant at every depth, but shrinks from -7.4 points at $K{=}10$ to -4.6 points at $K{=}2$. The tradeoff is real on DS-1000, but only its size is tunable. Table C.1 gives every depth against the base on both metrics, with the intervals these statements are read from.

Table C.1: Change in pass@1+ and pass@100+ compared to the base model for each K with 95% paired bootstrap confidence intervals over problems. † indicates a statistically significant difference. The sweep holds training at three epochs rather than the validation-selected protocol of the reported runs (Appendix B), so its $K{=}5$ row differs from Section 4.1.

Benchmark	K	Δpass@1+	Δpass@100+
HumanEval+	2	-0.8 $[-3.6, +2.0]$	$+7.9$ † $[+4.2, +12.2]$
	3	-1.3 $[-4.2, +1.6]$	$+10.2$ † $[+6.0, +14.8]$
	5	-1.7 $[-4.9, +1.6]$	$+10.4$ † $[+6.1, +15.2]$
	7	-1.5 $[-5.1, +2.2]$	$+10.5$ † $[+6.0, +15.1]$
	10	-3.1 $[-7.4, +1.1]$	$+9.3$ † $[+4.8, +14.3]$
MBPP+	2	$+0.9$ $[-1.3, +3.3]$	$+10.2$ † $[+7.4, +13.3]$
	3	$+0.1$ $[-2.6, +2.6]$	$+11.0$ † $[+7.8, +14.5]$
	5	-2.6 $[-5.4, +0.2]$	$+12.3$ † $[+9.0, +15.9]$
	7	-3.0 † $[-5.9, -0.0]$	$+11.0$ † $[+7.5, +14.7]$
	10	-4.4 † $[-7.5, -1.3]$	$+9.9$ † $[+6.6, +13.6]$
DS-1000	2	-4.6 † $[-6.2, -3.2]$	$+10.1$ † $[+7.8, +12.3]$
	3	-5.5 † $[-7.1, -3.9]$	$+11.1$ † $[+8.7, +13.4]$
	5	-6.8 † $[-8.7, -5.0]$	$+10.9$ † $[+8.4, +13.5]$
	7	-6.7 † $[-8.6, -4.9]$	$+9.6$ † $[+7.0, +12.1]$
	10	-7.4 † $[-9.4, -5.5]$	$+8.0$ † $[+5.4, +10.5]$

The reported runs use $K{=}5$. It sits at the top of the coverage plateau and well short of the depth where the cost to pass@1+ accelerates, whereas $K{=}10$ is the weakest setting on both axes: it gives up the most single-sample accuracy and never buys additional coverage. One confound remains. The sweep varies K, which varies the conditioning depth and the number of targets together, so the decline in pass@1+ is equally consistent with conditioning depth or with drift under more optimizer updates; on HumanEval+, the $K{=}10$ arm takes roughly 308 optimizer updates, 154 at $K{=}5$, and 62 at $K{=}2$, and the matched i.i.d. SFT control is a depth-one arm. The sweep also holds training at three epochs, so longer chains are trained longer; the validation-selected protocol of the reported runs (Appendix B) removes this but does not retroactively disentangle the sweep. Separating the two would require varying the chain length at a fixed total number of targets, which we have not run. Target quality by chain position is in Appendix D.

D TARGET QUALITY ALONG THE CHAIN

Later positions pass less often, with no clean trend. Figure D.1 reports the pass rate at each position of a chain of length $K{=}20$. It is 0.787 at position 1 and 0.616 at position 20, with substantial non-monotonic variation in between. The lower endpoint is consistent with a tradeoff between differing from prior solutions and correctness, but it does not establish that the quality of later positions causes the $K{=}5$ optimum of Appendix C.

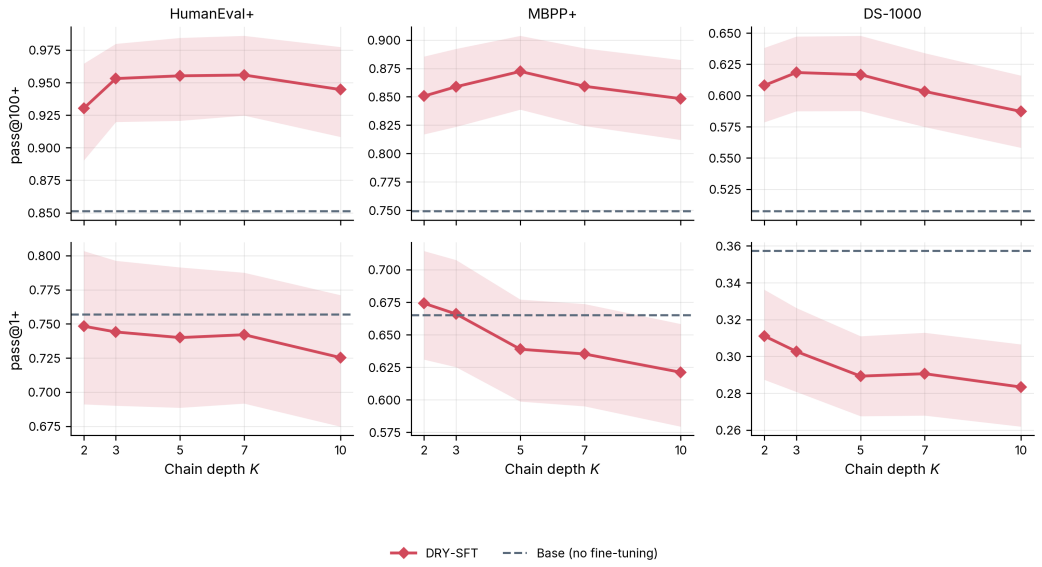


Figure C.1: pass@100+ (**top**) and pass@1+ (**bottom**) for different K. Shaded regions are 95% bootstrap confidence intervals over problems; dashed lines show the base model. We choose K=5 for fine-tuning. Appendix C reports the change compared to the base model.

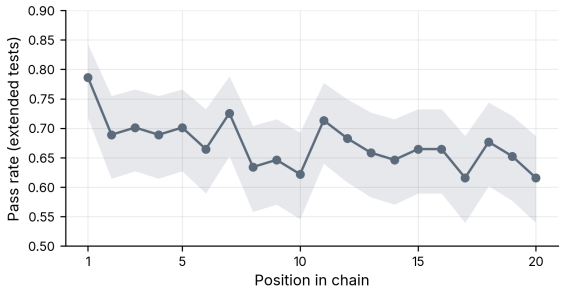


Figure D.1: Pass rate at each position in a chain of K=20 solutions, evaluated on the extended test suite. Shaded regions show 95% confidence intervals.

E CORRECTNESS FILTERING BEFORE STAGE 2

We compare default DRY-SFT with correctness filtering after Stage 1 and before Stage 2. The unfiltered arm is the held-out DRY-SFT model of Section 4.4. The filtered arm is retrained from Qwen3-4B-Instruct-2507 using the same K=5 chained candidates and seed-0 50/50 task splits. The filtered arm retains only targets passing both the original and extended tests on HumanEval+ and MBPP+, or the native tests on DS-1000. Filtering applies to training and validation targets, with validation task membership fixed before filtering. Stage 1 is unchanged: earlier failed attempts remain in the context for subsequent generations. The filtered arm uses the same training seed, optimizer settings, five-epoch budget, and minimum-validation-loss checkpoint selection as the held-out experiment. We evaluate on the untouched task half with 200 independent samples per problem at T=0.8 and top-p=0.95.

Filtering changes coverage little. Figure E.1 shows small differences between the two variants. At k=100, filtering changes coverage by +0.1, −0.6, and +3.2 percentage points on HumanEval+, MBPP+, and DS-1000, respectively (Table E.1). The paired intervals include zero on HumanEval+ and MBPP+, but exclude zero in favor of filtering on DS-1000. At k=1, filtering improves all three benchmarks with intervals excluding zero.

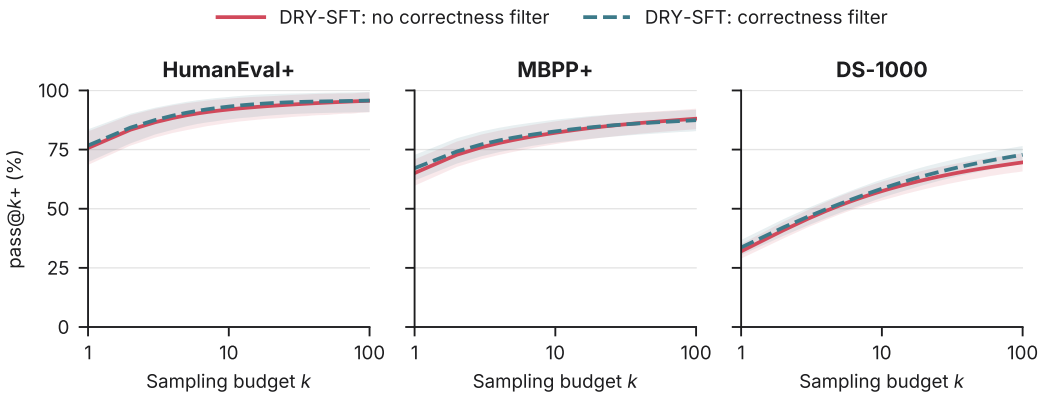


Figure E.1: Held-out pass@k+ with and without correctness filtering before Stage 2. Both arms use the same generated targets before filtering and the same dataset split. Shaded regions are pointwise 95% bootstrap confidence intervals over problems. Appendix E reports paired differences.

Table E.1: Correctness-filtering ablation. Coverage is reported as a percentage; Δ is filtered minus unfiltered in percentage points, with pointwise 95% paired problem-bootstrap intervals from 2,000 resamples. Intervals quantify held-out task uncertainty for one training seed per arm. Values are rounded to one decimal place.

Benchmark	k	Unfiltered	Filtered	Δ [95% CI]
HumanEval+	1	75.7	76.7	+1.1 [+0.2, +2.0]
	100	95.6	95.7	+0.1 [−1.7, +1.9]
MBPP+	1	65.1	67.1	+2.0 [+0.1, +4.1]
	100	88.1	87.4	−0.6 [−2.4, +1.0]
DS-1000	1	32.0	33.6	+1.6 [+0.1, +3.2]
	100	69.6	72.8	+3.2 [+0.8, +5.8]

The coverage gains need neither filtering nor failed targets. Filtering retains 276/410, 586/945, and 689/2500 candidate targets, including validation targets, on HumanEval+, MBPP+, and DS-1000. We do not replace discarded targets, so filtering also changes task coverage, optimizer-update counts, validation targets, and potentially the selected epoch. Both variants retain substantial coverage gains over the held-out base model (Section 4.4). These gains require neither correctness filtering nor fitting test-failing Stage 2 targets. This comparison does not isolate the effect of incorrect targets at a fixed training budget, rule out a role for failed attempts in Stage 1 conditioning, or establish equivalent performance across training seeds.

F Per-model coverage curves

Section 4.5 reports how a base model's structural diversity relates to the gain DRY-SFT produces across nine models. Figure F.1 gives the underlying pass@k+ curves, one per model, each against its own matched i.i.d. SFT control. The models are Qwen3-8B in its no-thinking mode (Qwen Team, 2025), EXAONE-3.5-7.8B-Instruct (LG AI Research, 2024), Qwen2.5-7B-Instruct (Qwen Team, 2024), granite-3.3-8b-instruct (Granite Team, IBM, 2025), OLMo-2-7B-Instruct (Team OLMo et al., 2024), Ministral-8B-Instruct-2410 (Mistral AI, 2024), and deepseek-coder-6.7b-instruct and deepseek-coder-1.3b-instruct (Guo et al., 2024). Every model follows the same protocol as the primary model: K=5, 5 unfiltered targets per problem, and no reward, verifier, or correctness filter at any point.

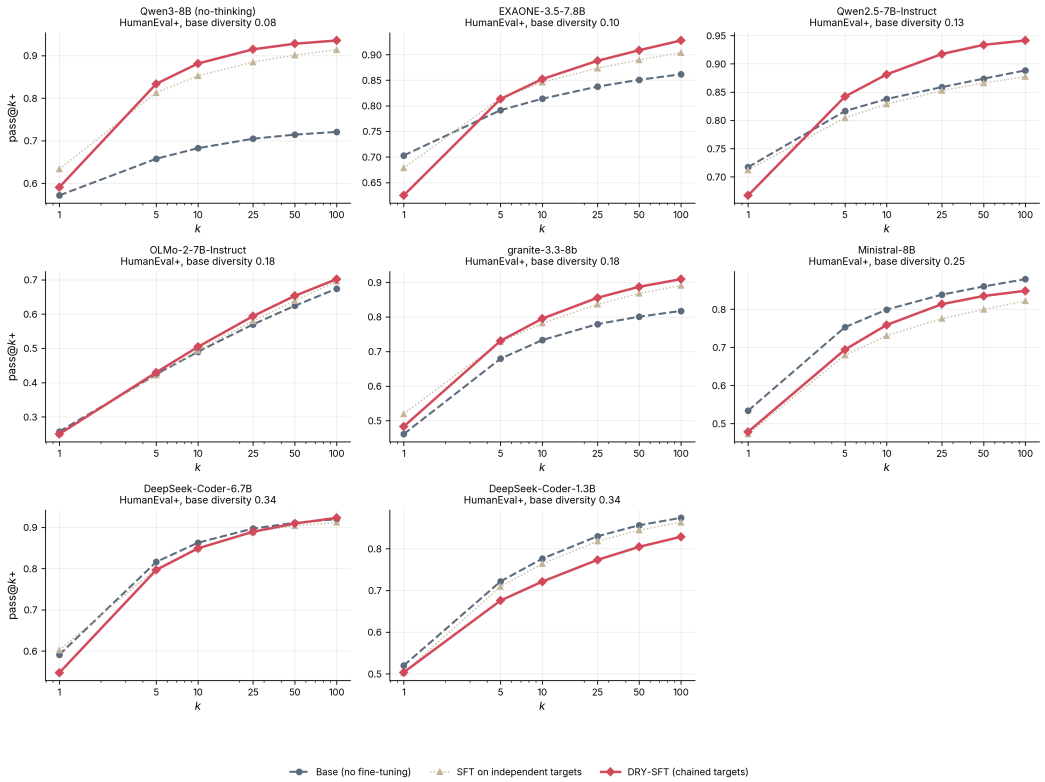


Figure F.1: HumanEval+ pass@k+ for each model, comparing DRY-SFT with its base model and matched i.i.d. SFT control. Models with lower base structural diversity are shown first. Before evaluation, we remove output that is not code.

G PROMPT TEMPLATES

The exact prompts used in Stage 1 are different across benchmarks, as they need to match the output format of each benchmark. The instruction to differ from every prior solution is shared across benchmarks. We reproduce the templates below, as they appear in the generation code, before substituting placeholders. Each box holds one benchmark's system prompt above the divider and its user templates below, the first-position template being the one Stage 2 and evaluation reuse.

HumanEval+ Stage 1

SYSTEM PROMPT

```
You are a Python coding assistant that generates diverse solutions.
When shown previous solutions to a problem, you produce a new
solution that uses a fundamentally different algorithm or approach.
```

USER PROMPT

FIRST POSITION

```
Implement the following Python function.
Output ONLY the function body. No explanation, no markdown.

{prompt}
```

LATER POSITIONS

```
Here is a Python function to implement:

{prompt}

The following solutions have already been produced:

{previous_solutions}

Write a NEW solution that uses a DIFFERENT algorithm or approach from
 all of the above. Output ONLY the function body. No explanation, no
markdown.
```

For HumanEval+, we serialize each prior target as `--- Solution {i} ---` followed by the complete function prompt and generated body.

MBPP+ Stage 1

SYSTEM PROMPT

```
You are a Python coding assistant that generates diverse function
implementations. When shown previous solutions to a problem, produce
a new solution that uses a different approach. Output ONLY full
Python function code, no markdown.
```

USER PROMPT

FIRST POSITION

```
Solve the following MBPP task.
Output ONLY the full Python function definition needed to solve it.

{prompt}
```

LATER POSITIONS

```
Solve the following MBPP task:

{prompt}

The following solutions have already been produced:

{previous_solutions}

Write a NEW solution that uses a DIFFERENT algorithm or approach from
 all above. Output ONLY full Python function code. No markdown.
```

For MBPP+, we serialize each prior target as `--- Solution {i} ---` followed by the generated full function.

DS-1000 Stage 1

SYSTEM PROMPT

```
You are a data-science coding assistant that generates diverse
completions. When shown previous completions, produce a new
completion that uses a fundamentally different approach. Output ONLY
the completion code in a ```python``` block.
```

USER PROMPT

FIRST POSITION

```
Complete the following data-science coding problem.
Output ONLY the missing code completion.

{prompt}
```

LATER POSITIONS

```
Complete the following data-science coding problem:

{prompt}

The following completions have already been produced:

{previous_solutions}

Write a NEW completion that uses a DIFFERENT approach from all above.
 Output ONLY the missing code completion.
```

For DS-1000, each prior target is serialized as a numbered solution with a fenced Python code block.

Stage 2 and evaluation. For every example in Stage 2, we pair the generated target with the bare benchmark problem and a neutral correctness-format system prompt, without any prior solution or diversity prompt. The user side is the first-position template of the matching benchmark above; only the system prompt changes, and MBPP+ is the one benchmark whose evaluation prompt differs from the one it was trained under.

HumanEval+ Stage 2 and evaluation

SYSTEM PROMPT

```
Complete the following Python function. Output ONLY the function body
 (the code that goes inside the function). Do not repeat the function
 signature or docstring. Do not include any explanation or markdown
formatting.
```

MBPP+ Stage 2

SYSTEM PROMPT

```
Write a correct Python function implementation. Output ONLY the full
function definition. No explanation and no markdown.
```

MBPP+ evaluation

SYSTEM PROMPT

```
Write a correct Python function implementation. Output ONLY valid
Python code for the full function definition. No explanation and no
markdown.
```

DS-1000 Stage 2 and evaluation

SYSTEM PROMPT

```
You are an expert data scientist writing Python code. Complete the
unfinished code. Output ONLY the code that should replace the
incomplete portion, preferably in a ```python``` block. Do not repeat
 the provided context.
```