NANOSWE: SPEEDRUNNING SWE-BENCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Speedrunning has become a popular engineering challenge in machine learning: achieve the best performance within a fixed compute budget. We use it as a scientific tool to study how models acquire specialized skills and how far those skills transfer. We speedrun SWE-bench by training coding agents from scratch on demonstrations from stronger models. With 192 B200-hours of training compute, we reach 25.5% pass@1 on SWE-bench Verified, outperforming GPT-4o. Scaling laws fitted to smaller runs forecast this result to within 0.3 percentage points. At matched pass@1, our agents solve largely the same SWE-bench problems as fine-tunes of a model pre-trained on trillions of tokens, and generalize about as well to new bugs and codebases. Beyond SWE-bench, however, our speedrun's capabilities are strikingly jagged: on MMLU, they do worse than GPT-2. What, then, does pre-training buy for SWE-bench? Controlled experiments support a one-student hypothesis: pre-training helps a student imitate its teacher, but once we know how well it does so, how it was pre-trained barely matters. The hypothesis holds even when comparing a model trained on the web against a "vintage" one trained only on text from before 1931. Up to the SWE-bench scores we reach, pre-training is interchangeable with specialization data. We release nanoswe as an open project for studying specialization through speedrunning.

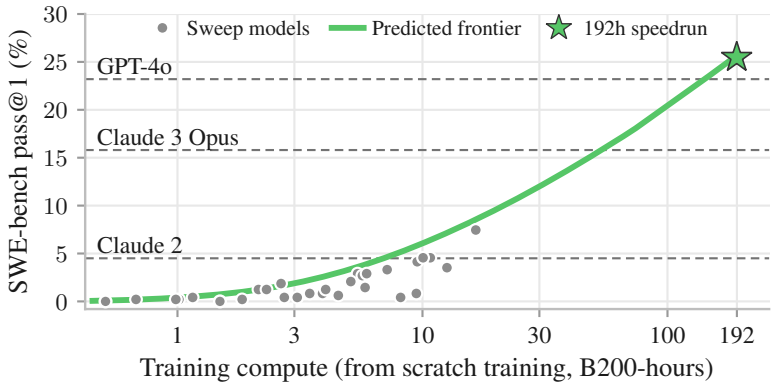


Figure 1: We speedrun SWE-bench by training from scratch on teacher demonstrations. Performance scales predictably: from a 105 B200-hour scaling sweep (•), we forecast the best pass@1 reachable at each compute budget with our training setup (—). The forecast holds at 192 B200-hours: we reach 25.5% pass@1 (★), outperforming GPT-4o and Claude 3 Opus for about $1,000 in training compute.

1 INTRODUCTION

Speedruns turn training efficiency into a competition, with remarkable success. The DAWNBench challenge (Coleman et al., 2017) cut the time to train a model to 93% top-5 accuracy on ImageNet from over 10 days to under 30 minutes. In language modeling, modded-nanogpt (Jordan et al., 2024a) has seen more than 100 records in two years, cutting the time to reach 3.28 validation loss on FineWeb from 45 minutes to 68 seconds. Along the way, speedruns have produced a wealth of engineering know-how, such as the Muon optimizer (Jordan et al., 2024b) we use to train our models.

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.

In this work, we turn speedrunning into a scientific instrument for studying model specialization. Racing to a capability strips training down to what matters. We speedrun SWE-bench (Jimenez et al., 2024; OpenAI, 2024), the benchmark that has charted the rise of coding agents. When it appeared at ICLR 2024, the state of the art solved only 5% of what is now its Verified subset. Today SWE-bench is widely considered solved (OpenAI, 2026), and coding agents are a multibillion-dollar business (Anthropic, 2026; Zeff, 2026).

We set out to find just how efficiently we can turn training compute into SWE-bench performance. We start from randomly initialized weights and constrain one resource: wall-clock time on a single 8×B200 node. A speedrun may train for at most one day (192 B200-hours, ≈$1,000), but we place no limit on the cost of data generation, experimentation, or evaluation. Our starting question, then, is simple: how good a coding agent can we train in a day?

1.1 Our contributions

Training from scratch, mostly on teacher demonstrations, our one-day speedrun reaches 25.5% pass@1 on SWE-bench Verified, outperforming Claude 3 Opus and GPT-4o (Figure 1). Pass@1 emerges after just 0.4 B200-hours and grows predictably: scaling laws fitted on 29 smaller runs, 105 B200-hours in total, forecast 25.2% for our speedrun, an elevenfold extrapolation (Figure 2).

Cheap as they are, our speedruns buy the same kind of SWE-bench capability as models with extensive general pre-training. To see this, we compare against fine-tunes of SmolLM3, a model pre-trained on 11 trillion tokens. For matched pass@1, our speedruns and SmolLM3 fine-tunes largely solve the same SWE-bench problems (Figure 3). And on four new test sets of recent and synthetic bugs, speedruns generalize about as well as the SmolLM3 fine-tunes (Figure 4). Beyond SWE-bench, however, the capabilities of our speedruns are strikingly jagged: the one-day speedrun beats GPT-4o on SWE-bench, yet does no better than GPT-2 on MMLU (Figure 5).

What, then, does pre-training buy for SWE-bench? We test a simple answer, the *one-student hypothesis*: pre-training makes the teacher easier to imitate, but confers surprisingly little task competence beyond what the model's imitation loss, its loss on held-out teacher demonstrations, already captures. We fine-tune three models with very different pre-training (SmolLM3 base, SmolLM3 mid-trained, and Qwen 3 4B) on demonstrations from two teachers. At matched imitation loss, their SWE-bench abilities converge within about a billion fine-tuning tokens (Figure 6). Once we know the models' imitation loss, how they were pre-trained is largely irrelevant. The same view explains why, depending on the student, a stronger teacher can produce a weaker student (Figure 7).

Finally, we stress-test the one-student hypothesis by comparing a model pre-trained on the web with a "vintage" model pre-trained only on documents from before 1931. The hypothesis holds up: a century of distribution shift can be offset by training on just 3× more specialization data (Figure 8).

1.2 Related work

Speedruns race to a fixed target, such as an accuracy on ImageNet (Coleman et al., 2017) or a validation loss on FineWeb (Jordan et al., 2024a; Karpathy, 2025); others instead fix train budget as we do (Geiping & Goldstein, 2023). We target SWE-bench, and place no limit on the data synthesis budget. Whereas open-weight coding agents are typically built by fine-tuning strong pre-trained models on agent trajectories generated by stronger teachers (Pan et al., 2025; Jain et al., 2025b; Yang et al., 2026), we uniquely start from randomly initialized model weights.

Scaling laws for transfer treat pre-training as extra fine-tuning data or as a factor on the fine-tuning loss (Hernandez et al., 2021; Zhang et al., 2024a; Barnett, 2024). Models with equal pre-training loss can perform alike downstream (Du et al., 2024) or not (Liu et al., 2023); we find that after about a billion fine-tuning tokens, pre-training adds no ability beyond what imitation loss captures. The same finding predicts when a stronger teacher yields a weaker student. Prior work attributes this to a capacity gap (Cho & Hariharan, 2019; Busbridge et al., 2025; Li et al., 2025); our students have similar sizes, and what we find to matter is their pre-training.

Fine-tuning on a task makes seemingly emergent abilities predictable from smaller models (Snell et al., 2024; Dominguez-Olmedo et al., 2025); our speedruns train on the task from the start. Rather than mapping loss to accuracy (Gadre et al., 2025), we map it to latent ability (Truong et al., 2026).

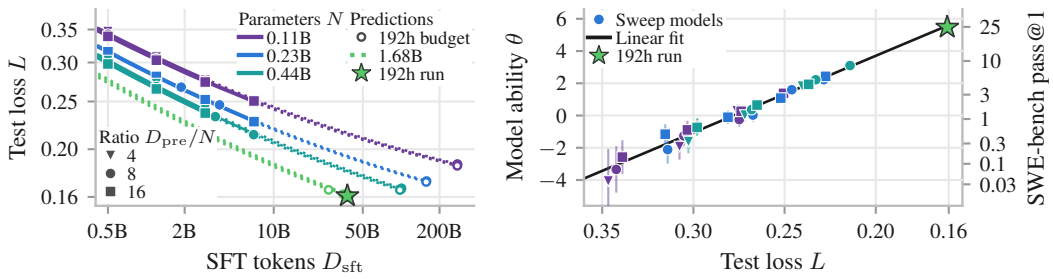


Figure 2: Scaling laws fitted on our 29-model sweep. *Left*: Student-on-teacher perplexity follows a Chinchilla-like law. *Right*: Latent model ability scales linearly with test loss, yielding pass@1 predictions via a Rasch model starting at 0.4 GPU-hours of train compute.

Our new test sets follow work that re-collects data to detect overfitting (Recht et al., 2019; Mirzadeh et al., 2025). See Appendix J for additional related work.

2 Speedrunning SWE-bench

2.1 Predictable scaling

To allocate our speedrun budget efficiently, we first characterize how SWE-bench performance scales with model size and training tokens through a 29-model sweep. We train models from scratch using the nanochat (Karpathy, 2025) training stack, with minor modifications to improve long-context training throughput. Training proceeds in two phases: pre-training on FineWeb Edu (Lozhkov et al., 2024), followed by supervised fine-tuning on up to 1.4 million agentic trajectories (33.2B tokens) generated by Nemotron 3.5 Lightning (Blakeman et al., 2025), see Appendix A. We ensure that none of our training data originates from the SWE-bench codebases.

Our sweep covers model depths {12, 16, 20}, corresponding to N ={110M, 235M, 435M} parameters; pre-training token-to-parameter ratios D_{pre}/N={4, 8, 16}; and fine-tuning budgets D_{sft}={0.5B, 1.2B, 2.9B, 7.0B} tokens. In total we train 29 models at a combined cost of 105 B200 hours, with individual runs ranging from 0.4 to 17 B200 hours. See Appendix B for details.

Perplexity scaling. We generate Nemotron rollouts on the SWE-bench instances and measure each model's perplexity L (bits per byte) on these held-out rollouts. We find that student-on-teacher perplexity follows a Chinchilla-like law (Hoffmann et al., 2022) in N and D_{sft},

$$L = E + AN^{-\alpha} + BC_{\text{pre}}^{-\gamma} D_{\text{sft}}^{-\beta}, \tag{1}$$

where the amplitude of the SFT term decreases with pre-training compute $C_{\text{pre}} = 6ND_{\text{pre}}$. We constrain the irreducible error E to be no lower than the teacher's own perplexity on its trajectories.

We fit the coefficients by least squares ($E = 0.066$, $A = 0.0239$, $\alpha = 0.450$, $B = 0.174$, $\gamma = 0.0192$, $\beta = 0.212$), achieving a root mean squared relative error of 0.5% (Figure 2 left).

Pass@1 scaling. We relate test loss L to pass@1[1] by building on Item Response Scaling Laws (Truong et al., 2026), which model benchmark performance through a latent ability parameter. Whereas they regress ability on training compute, we regress it on test loss.

Our law has two steps. First, we fit a Rasch model (Rasch, 1993) via maximum likelihood,

$$p_{m,i} = \sigma(\theta_m - z_i), \tag{2}$$

where $p_{m,i}$ is the solve rate of model m on SWE-bench instance i, θ_m is model ability, z_i is instance difficulty, and σ is the logistic function. Second, we regress model ability on test loss via WLS,

$$\theta = \theta_0 - \kappa L. \tag{3}$$

[1]Since our models are relatively weak, we improve the precision of our pass@1 estimates by splitting SWE-bench into an easy subset (the 100 instances with the highest solve rate in mini-coder's (Olmedo et al., 2026) pass@k data) and a hard subset, sampling 5 attempts on the easy subset and one on the hard subset.

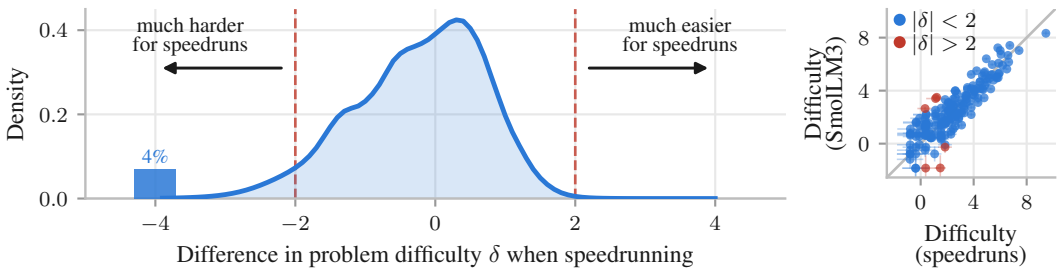


Figure 3: Per-problem difficulty differences δ (Eq. 4) between 21 from-scratch speedruns and 69 SmolLM3 fine-tunes on SWE-bench. The two cohorts largely agree on which problems are hard.

At inference time, we predict pass@1 for a given test loss L by chaining Equations 3 and 2 and averaging over all SWE-bench instances. We find that pass@1 is well fitted by this loss-to-performance law (Figure 2 right). Remarkably, model ability falls on a single line against test bpb regardless of model size, pre-training budget, or fine-tuning budget. That is, student-on-teacher loss alone captures all variability in model ability across our sweep.

Even our smallest run, just 0.4 B200-hours of training compute, achieves non-zero pass@1. This is surprising: one would expect agentic coding ability to emerge only at much larger scales (Srivastava et al., 2022; Wei et al., 2022; Ganguli et al., 2022). Yet after a small amount of distillation, even tiny models show measurable ability, which grows predictably. For example, we predict the post-distillation pass@1 of SmolLM3 from its early pre-training checkpoints (Appendix C).

2.2 The efficiency frontier

Our scaling laws let us trace an *efficiency frontier*, i.e., the maximum pass@1 our setup can attain for a given training budget. Since pass@1 is monotonic in test loss, we compute this frontier by minimizing L over $(N, D_{\text{pre}}, D_{\text{sft}})$ via Equation 1, then mapping the result through the pass@1 law. We plot the estimated efficiency frontier in Figure 1, extrapolating up to 192 GPU-hours (a full day on an 8×B200 node, or 12× the sweep's largest run).

To validate this extrapolation, we train a 192-hour speedrun using the implied optimal configuration: $N^* = 1.68$B (depth 32), $D^*_{\text{pre}}/N^* = 4$, and $D^*_{\text{sft}} = 37.8$B tokens. As shown in Figure 2, the speedrun achieves a test loss of 0.161 bpb (predicted: 0.162) and 25.5% pass@1 (predicted: 25.2%). Both values match our laws' predictions to within one percentage point.

To put this result in perspective, our speedrun outperforms both Claude 3 Opus (15.6% pass@1) and GPT-4o (23.2% pass@1)[2]. Once teacher demonstrations are readily available for training, about $1000 worth of compute takes us to the 2024 SWE-bench state-of-the-art. Teacher demonstrations thus allow us to buy SWE-bench performance surprisingly cheaply, requiring little general pre-training. We next ask what, if anything, we lose by taking this shortcut.

3 What does speedrunning buy?

Our speedruns train primarily on teacher demonstrations, with little general pre-training. The standard paradigm is instead to pre-train at scale, and only then post-train. In Section 3.1, we first ask whether skipping pre-training changes which SWE-bench instances models find difficult. In Section 3.2, we then ask whether forgoing pre-training hurts generalization to new bugs and codebases at comparable SWE-bench performance. Finally, in Section 3.3, we move past agentic coding and evaluate our speedruns on tests of general knowledge and mathematical reasoning.

[2]As evaluated by the official SWE-bench leaderboard using SWE-Agent, a scaffold comparable to ours.

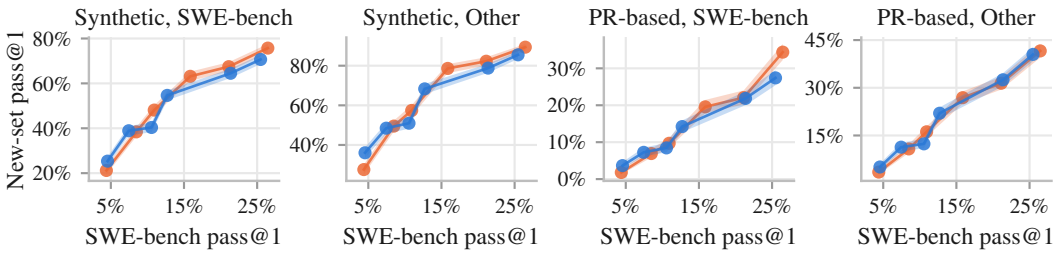


Figure 4: Pass@1 on four novel test sets vs. SWE-bench Verified pass@1, for **from-scratch speedruns** (●) and **SmolLM3 fine-tunes** (●) trained on the same teacher demonstrations. Panels vary bug source and codebase. At matched Verified score, the speedruns trail on average by 1.4, 0.6, 0.8, and −0.3pp, respectively (Equation 5), with all four 95% intervals including zero.

3.1 Does skipping web pre-training change which problems are hard?

Skipping web-pretraining could bias which problems our speedruns are able to solve. Intuitively, some SWE-bench instances may become much harder without certain knowledge present in the web, such as libraries' documentation, Stack Exchange discussions, or GitHub issue threads.

We contrast two cohorts of models: 21 models trained from scratch *only* on teacher demonstrations, and 69 SmolLM3 3B fine-tunes. For the latter, we vary both the pretraining budget (from 100B to 10T tokens) and the fine-tuning budget (from 100M to 5B tokens). Both the from scratch models and the fine-tunes train on the same teacher demonstrations, mini-coder-trajs-400k (Olmedo et al., 2026). The two model cohorts thus differ mainly in SmolLM3's extensive web and code pre-training. See Appendix F for additional training details. We then evaluate each model on SWE-bench with up to k=90 attempts per problem.

We ask whether these two cohorts, A and B, agree on which SWE-bench problems are easy. To answer this, we borrow a standard methodology from psychometrics: differential item functioning (DIF; Holland & Wainer, 1993). We fit a single Rasch model with DIF to all outcomes,

$$\operatorname{logit} p_{ij} = \theta_i + z_j + c_i\,\delta_j, \tag{4}$$

where p_{ij} is the probability that model i solves problem j on a single attempt, θ_i is the ability of model i, z_j the easiness of problem j, and c_i indicates membership in cohort A. Our object of study is δ_j: how much easier problem j is for cohort A than for cohort B, at matched model ability θ. We estimate and plot in Figure 3 the distribution over δ, with each problem contributing one unit of probability mass, spread according to its measurement uncertainty.[3] We mark $\delta = \pm 2$ in each panel as the threshold at which the two cohorts substantially disagree on problem difficulty.[4]

As shown in Figure 3 (left), the two cohorts largely agree on problem difficulty: the distribution of δ is unimodal, with most of its mass close to zero. Put differently, at the same overall pass@1, both cohorts tend to solve the same SWE-bench problems. The exception is a small negative tail of problems that are substantially harder for the speedruns. Whatever latent knowledge the SmolLM fine-tunes gain from web pre-training seems to surface in only a handful of problems. By contrast, against the public SWE-bench leaderboard, our speedruns find far more problems much harder (Appendix H).

3.2 Generalization to new bugs and codebases

We next ask whether forgoing large-scale pre-training hurts generalization to new bugs and codebases, again using SmolLM3 fine-tunes as a reference point. We build four new test sets, varying the bug source (LM-introduced synthetic bugs or bugs fixed by recent GitHub pull requests) and the codebase (SWE-bench or "other" codebases).

[3] Our analysis determines δ only up to an additive constant; we anchor the scale by setting the median δ to zero (Kopf et al., 2015; Bechger & Maris, 2015). A uniform benefit of web pre-training across items is absorbed by the anchor; what we set out to detect is difficulty reranking. The figure shows the 199 items for which the data at least pins down δ's sign (only problems solved by either cohort carry information).

[4] z has standard deviation of about 3 logits; $|\delta| < 2$ moves the overall difficulty ranking only modestly.

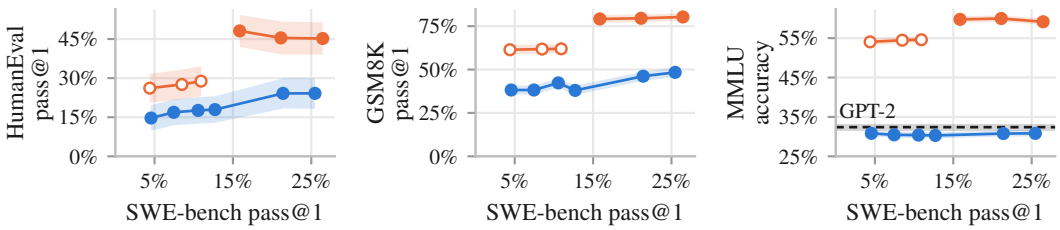


Figure 5: HumanEval, GSM8K, and MMLU vs. SWE-bench pass@1 after identical fine-tuning on a shared elicitation dataset, for **from-scratch speedruns** (•) and **SmolLM3 fine-tunes** starting from the base (○) or mid-trained (•) checkpoint. Our speedruns perform much worse.

We obtain synthetic bugs by prompting Opus 4.8 to inject relatively small, self-contained bugs into a target codebase. For pull-request bugs, we instead mine GitHub for bug fixes merged in 2026, and revert them. Synthetic bugs are guaranteed unseen but artificial; pull-request bugs are realistic but public. We use SWE-smith to build the execution environments and verify that each bug is caught by its codebase's unit tests.

We use 23 repositories from SWE-bench and 28 "other" GitHub repositories with a similar star distribution (median ≈17k stars); see Appendix E for full details. All test codebases are strictly held out from our training data, but many appear in SmolLM3's pre-training corpus.

We compare six from-scratch speedruns, trained on 3B to 38B tokens, against six SmolLM3 fine-tunes, trained on 200M to 2.2B tokens. Both cohorts train on Lightning demonstrations (Appendix G), so the main difference between them is SmolLM3's 11T tokens of pre-training. We evaluate all twelve models on SWE-bench Verified and the four new test sets. If pre-training aids generalization, SmolLM3 should score higher on the new test sets at the same Verified score.

Figure 4 plots each model's new-test pass@1 against its Verified pass@1. Across all four test sets, performance on the new test sets increases with Verified performance, and the two cohorts follow broadly similar trends. To quantify the difference between them, we fit to each panel

$$\text{pass@1}_{\text{new}} = a + b\,\text{pass@1}_{\text{Verified}} + c_i\,\gamma, \tag{5}$$

where c_i indicates a SmolLM3 fine-tune, so that γ is the additional pass@1 of SmolLM3 on the new test set at the same Verified score. We fit a, b, and γ by least squares, with γ estimated at $+1.4$, $+0.6$, $+0.8$, and -0.3pp for the four panels from left to right. Bootstrapping over instances of the new test sets gives 95% intervals of ±1 to ±2 points around each estimate, all of which include zero. On average, SmolLM3's advantage is thus small at best.

Individual models vary more: SmolLM3's lead grows at higher scores[5], reaching +5.7pp on PR-bugs in SWE-bench codebases at around 25% Verified. SmolLM3 has seen these codebases during pre-training, so this gap may partly reflect more familiarity rather than better generalization.[6]

The gains of our speedruns thus extend beyond SWE-bench Verified to new bugs and codebases. Despite their lack of large-scale pre-training, our speedruns generalize about as well as the SmolLM3 fine-tunes, with an average deficit of at most 1.4 points per test set.

3.3 Beyond SWE-bench

So far, we have tested generalization within a very particular class of tasks: given a codebase and an issue description, implement a small, relatively self-contained fix. How much broader capability comes along with SWE-bench speedrunning? To find out, we evaluate our models on HumanEval (Chen et al., 2021) for non-agentic code generation, GSM8K (Cobbe et al., 2021) for mathematical reasoning, and MMLU (Hendrycks et al., 2021) for world knowledge.

[5]The three lowest-scoring SmolLM3 fine-tunes start from the base checkpoint, the three highest from the mid-trained one. Separate offsets give $\gamma = -1.5, -3.7, -1.0, -1.1$ and $+4.5, +5.4, +2.8, +0.4$pp, respectively; with three models each, none is significant once variation between models is accounted for.

[6]SmolLM3 trains on The Stack v2, which contains most of the repositories comprising SWE-bench.

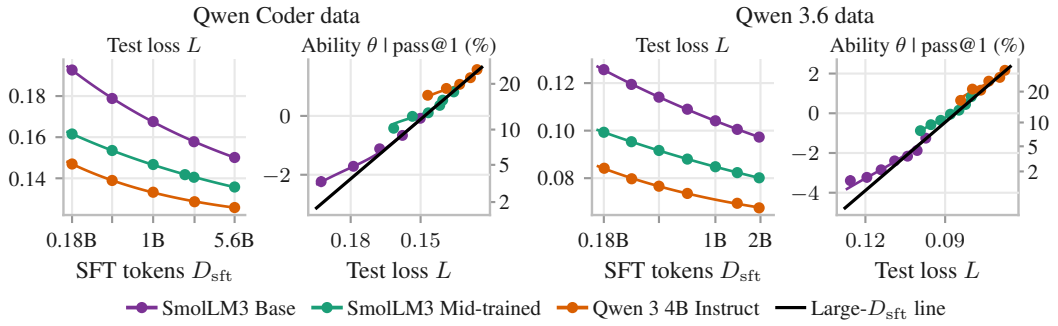


Figure 6: Fine-tuning three models on trajectories from two teachers, Qwen Coder (left pair) and Qwen 3.6 (right pair). Plotted against imitation loss, the abilities of all three models converge onto one line (black) within about 1B SFT tokens; colored curves show Equation 6.

We evaluate the same twelve models as in Section 3.2, now separating the SmolLM3 fine-tunes by starting checkpoint: base (8T tokens) or mid-trained (11T tokens). To ensure that differences in benchmark performance are not simply due to differences in task elicitation (Zhang et al., 2026), we fine-tune every model on the same 230M-token elicitation dataset (Appendix G) before evaluation.

Figure 5 plots HumanEval, GSM8K, and MMLU against SWE-bench pass@1. As our speedruns climb from 5% to 25% on SWE-bench, they gain about 10 points on HumanEval (to 24%) and GSM8K (to 48%). On MMLU, they do not improve at all. All six stay at 30–31%, barely above chance (25%). SmolLM3 fine-tunes with the same SWE-bench score do far better on all three benchmarks: those starting from the base checkpoint lead our speedruns on average by 11, 23, and 24 points, and those starting from the mid-trained checkpoint by 23, 34, and 29 points.

The capabilities of our speedruns are thus strikingly jagged. On SWE-bench, our $1,000 speedrun outperforms GPT-4o. Yet on MMLU, we do no better than GPT-2 (Hendrycks et al., 2021).[7] Such jaggedness is the cost of buying SWE-bench performance so cheaply.

Speedrunning SWE-bench does not buy MMLU ability. Conversely, MMLU ability is not needed to reach 25% on SWE-bench. What, then, does pre-training buy for SWE-bench?

4 The One-Student Hypothesis

Pre-training famously makes specialization cheaper (Erhan et al., 2010; Howard & Ruder, 2018; Hernandez et al., 2021). But how does it help? It might make specialization data easier to learn (e.g., a teacher easier to imitate), or it might also add ability beyond what imitation loss captures. The distinction matters. If it only makes the data easier to learn, pre-training is interchangeable with specialization data: more of it closes any gap, and loss alone predicts ability.

We call this possibility the *one-student hypothesis*. Formally, it states that imitation loss, the loss on held-out specialization data, fully mediates the effect of pre-training on ability: once we know a model's imitation loss, how it was pre-trained no longer matters. To test it, we relate ability to imitation loss across three models of similar size but very different training.

The models are SmolLM3 base, pre-trained on 8T tokens of web and code; SmolLM3 mid-trained, trained on a further 3T tokens with an emphasis on synthetic data; and Qwen 3 4B Instruct 2507, pre-trained on over 30T tokens and post-trained by distillation from the flagship Qwen 3 models.

We pair these models with two specialization datasets of SWE-smith trajectories, each generated by a different teacher. One is mini-coder-trajs-400k, 5.6B tokens generated with Qwen 3 Coder 30B-A3B; the other is 2B tokens we generate with Qwen 3.6 35B-A3B using the same recipe. These experiments predate our Lightning data, but preliminary results with it agree (Appendix D).

We fine-tune each model on each dataset, with budgets from 180M tokens up to one epoch, and estimate each fine-tune's ability θ with the Rasch model of Section 2.1. Figure 6 shows the result.

[7] 30.8% vs. 32.4% after fine-tuning on the MMLU aux. training set, and 27.1% vs. 30.3% without.

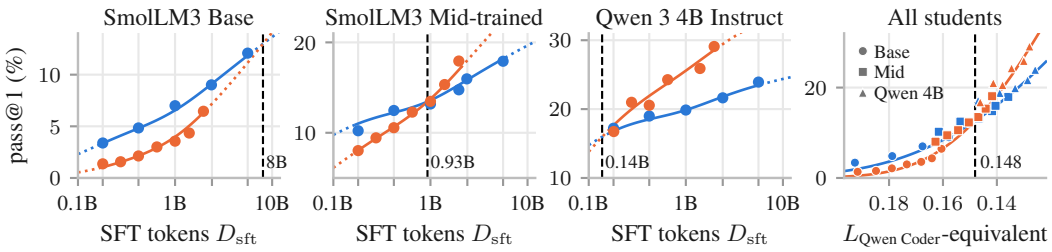


Figure 7: The better teacher depends on the student. *Left three:* pass@1 when trained on **Qwen Coder** (●) or **Qwen 3.6** (●); lines are Equation 6 (dotted: extrapolation; dashed: crossover). *Right:* fine-tunes against Qwen-Coder-equivalent loss; the lines cross once, at 0.148.

Pre-training clearly makes the data easier to learn: at every budget, the more heavily trained models reach a lower imitation loss (left panels). Yet plotted against imitation loss, the abilities of all three models converge onto a single line (right panels).

Early in fine-tuning, models solve more problems than their imitation loss implies. In this regime, a model's prior training does matter beyond how well it imitates the specialization data. This effect, however, vanishes surprisingly quickly. To measure how quickly, we extend the loss-to-ability law of Equation 3 with a discount on the loss that fades with fine-tuning,

$$\theta = \theta_0 - \kappa \left(1 - r\, e^{-D_{\text{sft}}/D_0}\right) L, \tag{6}$$

where the intercept θ_0 and slope κ are fitted per dataset, and the discount r and decay scale D_0 are shared across models and datasets. This law fits all 36 fine-tunes within measurement error: abilities deviate from it by 0.10 RMS, against standard errors of 0.075, or 0.49pp in pass@1. The discount is small and fades fast ($r = 0.10$, $D_0 = 0.35$B): each unit of imitation loss costs 6% less ability at 180M tokens, but only 0.5% at 1B, a difference smaller than our measurement error.

More tellingly, the law also has predictive power. If the one-student hypothesis holds, a law fitted on any two models should predict the third from its imitation loss alone. It does: held-out pass@1 errors are 0.5 to 0.9pp, with no systematic bias, even though at matched fine-tuning budgets, Qwen 3 4B and SmolLM3 base differ by 12 to 23pp.

Beyond one billion fine-tuning tokens, then, the three models are effectively one student: pre-training helps them reach a given imitation loss, but confers no ability beyond it. Next, we use this insight to explain why the stronger of our two teachers need not be the better choice.

4.1 When is a stronger teacher a better teacher?

Qwen 3.6 is a far stronger model than Qwen Coder on SWE-bench (58% vs. 28% pass@1). One might expect it to be the better teacher as well. Whether it is, however, depends on the student.

For our weakest student, SmolLM3 base, Qwen Coder is the better teacher at every budget we train (Figure 7). For SmolLM3 mid-trained, Qwen Coder is better at small budgets and Qwen 3.6 at large ones. For our strongest student, Qwen 3.6 is better at almost every budget.

The one-student hypothesis explains this pattern, and even predicts the budget at which Qwen 3.6 becomes the better teacher for each student. We need one more empirical fact: at matched budgets, every student's imitation losses obey $L_{\text{Qwen 3.6}} \approx 0.91\, L_{\text{Qwen Coder}} - 0.049$. Since ability depends on imitation loss alone, which teacher is better depends, roughly, only on how far a student has progressed, not on which student it is (Figure 7, right). The two teachers' curves therefore cross exactly once, at $L_{\text{Qwen Coder}} = 0.148$ and $L_{\text{Qwen 3.6}} = 0.086$ bpb, or about 13% pass@1: short of this point, students learn more from Qwen Coder; beyond it, from Qwen 3.6.

Why, then, is the weaker teacher better early on? The two teachers solve problems differently: Qwen Coder usually patches the code right away, whereas Qwen 3.6 first writes extensive tests to reproduce the issue. We speculate that weak students copy this test-first strategy before they can effectively execute it, get stuck writing tests, and often never produce a patch. Only once they imitate Qwen 3.6 closely enough does its thoroughness pay off.

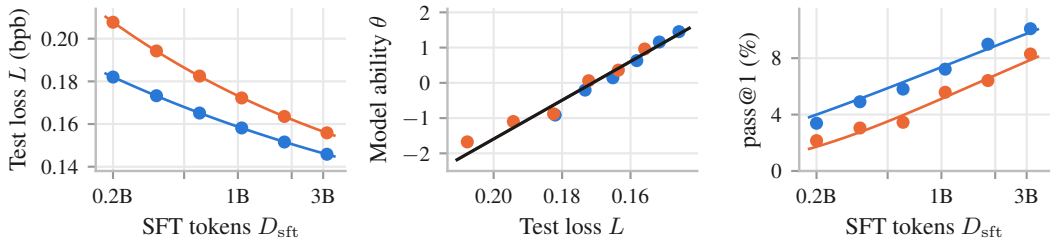


Figure 8: The Talkie 13B **web** (●) and **vintage** (●) models fine-tuned on mini-coder-trajs-400k. A single line (black) relates ability to imitation loss for both models (middle); chained with the loss fits (left), it reproduces SWE-bench pass@1 to within 0.36pp on average (right).

4.2 A CENTURY OF DISTRIBUTION SHIFT: PRE-TRAINING ON VINTAGE DATA

Every student we have studied so far was pre-trained on web text and code. We now stress-test the one-student hypothesis by comparing two models trained on radically different data: the Talkie "web" and "vintage" models (Levine et al., 2026). Both are Chinchilla-optimal 13B-parameter models trained identically, one on FineWeb (Penedo et al., 2024) and the other exclusively on pre-1931 documents. While the web corpus contains data directly relevant to SWE-bench issue fixing (e.g., GitHub and Stack Exchange), the vintage model's training cut-off predates the first electronic computer. If the one-student hypothesis holds, both models should nonetheless exhibit an identical relationship between imitation loss and SWE-bench ability.

We fine-tune both models on mini-coder-trajs-400k, the Qwen Coder dataset from above, with six log-spaced budgets from 0.2B to 3.2B tokens. As before, we estimate each fine-tuned model's SWE-bench ability, and plot it against imitation loss in Figure 8.

At matched SFT budgets, the web model is clearly the stronger student, solving 45% more problems on average (0.73 in ability). Yet plotted against imitation loss, both models fall close to a single line (Figure 8, middle).[8] Imitation loss thus accounts for nearly the entire gap: at matched loss, the web model solves at most 3% more problems (0.06 in ability, 95% bound).

The law also transfers between the two models: fitted on the web fine-tunes alone, it predicts the vintage model's pass@1 to within 0.54pp (16% relative) on average; fitted on the vintage fine-tunes, the web model's to within 0.45pp (9%). Chained with power-law fits of imitation loss against SFT budget, it reproduces the pass@1 of all 12 fine-tunes to within 0.36pp (8%).

Despite pre-training on radically different data, the web and vintage models are effectively one student too. To catch up, the vintage model only needs to match the web model's imitation loss, by training on between 2× and 3× the specialization data. For SWE-bench, and at Talkie's pre-training scale, surprisingly little seems to be lost by throwing away the internet.

5 CONCLUSION

Our one-day speedrun outperforms GPT-4o on SWE-bench with just $1,000 of training compute, despite doing worse than GPT-2 on MMLU. Yet at matched pass@1, our speedruns solve roughly the same problems as a model pre-trained on trillions of tokens, and generalize about as well to new bugs and codebases. We explain this with the one-student hypothesis: pre-training changes how quickly a student imitates its teacher, but at matched imitation loss, how it was pre-trained barely matters. It also explains why the stronger of two teachers need not be the better choice. And it holds even across radically different pre-training data: web text versus text from before 1931.

More broadly, our findings show how speedrunning can serve as an investigative tool for studying specialization. Speedruns thrive on competition, so we open-source our training stack, data, and models as nanoswe, a community SWE-bench speedrun. How far will one day of training take us?

[8] The law needs no early discount here: fitting Equation 6 with r fixed to its value above gives $D_0 = 0$.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, to give feedback on the design of experiments, and to help interpret results. We did not use them to develop theoretical models, formulate mathematical claims, propose hypotheses, clean or reformat datasets, or analyze qualitative data; proofs and translation are not applicable to this work. Our training data are synthetic by design: language models generate both the tasks and the agent trajectories we fine-tune on (Appendix A and Section 4.1). These data are the object of study rather than an aid to it, and we describe how we generate them in full. Additionally, we used generative AI tools to write and edit code, to create figures, to search for and summarize related literature, to draft parts of the related work, to format references, and to edit the paper for readability. We have reviewed all AI-assisted work, and we checked every bibliography entry against the published version of the cited work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. SWE-Bench+: Enhanced coding benchmark for LLMs. *arXiv preprint arXiv:2410.06992*, 2024.

Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning. *Transactions on Machine Learning Research*, 2022.

Anthropic. Anthropic raises $30 billion in Series G funding at $380 billion post-money valuation, February 2026. URL https://www.anthropic.com/news/anthropic-raises-30-billion-series-g-funding-380-billion-post-money-valuation. Accessed: 2026-09-25.

Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. To code or not to code? Exploring impact of code in pre-training. In *International Conference on Learning Representations*, 2025.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. *Advances in Neural Information Processing Systems*, 38, 2026.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Matthew Barnett. An empirical study of scaling laws for transfer. *arXiv preprint arXiv:2408.16947*, 2024.

Timo M Bechger and Gunter Maris. A statistical test for differential item pair functioning. *Psychometrika*, 80(2):317–340, 2015.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

David Brandfonbrener, Nikhil Anand, Nikhil Vyas, Eran Malach, and Sham Kakade. Loss-to-loss prediction: Scaling laws for all datasets. *Transactions on Machine Learning Research*, 2025.

Ryan Burnell, Han Hao, Andrew R. A. Conway, and Jose Hernandez Orallo. Revealing the structure of language model capabilities. *arXiv preprint arXiv:2306.10062*, 2023.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation scaling laws. In *International Conference on Machine Learning*, 2025.

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

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

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

Cody A. Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. DAWNBench: An end-to-end deep learning benchmark and competition. In *NIPS ML Systems Workshop*, 2017.

Fabrizio Dell'Acqua, Edward McFowland, Ethan Mollick, Hila Lifshitz, Katherine C. Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R. Lakhani. Navigating the jagged technological frontier: Field experimental evidence of the effects of artificial intelligence on knowledge worker productivity and quality. *Organization Science*, 37(2):403–423, 2026. doi: 10.1287/orsc.2025.21838.

Richard A. DeMillo, Richard J. Lipton, and Frederick G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, 1978. doi: 10.1109/C-M.1978.218136.

Ricardo Dominguez-Olmedo, Florian Eddie Dorner, and Moritz Hardt. Training on the test task confounds evaluation and emergence. In *International Conference on Learning Representations*, volume 2025, pp. 68564–68594, 2025.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, 2024.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208. JMLR Workshop and Conference Proceedings, 2010.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pp. 10421–10430, 2023.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. In *International Conference on Learning Representations*, 2025.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 1747–1764, 2022.

Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pp. 11117–11143. PMLR, 2023.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Paul W. Holland and Howard Wainer (eds.). *Differential Item Functioning*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance in machine translation. In *International Conference on Learning Representations*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025a.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Conference on Language Modeling*, 2025b.

Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011. doi: 10.1109/TSE.2010.62.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Jina AI. code_exercises. https://huggingface.co/datasets/jinaai/code_exercises, 2023. Dataset of ∼120k synthetic Python docstring-to-function exercises (*phi-1* / "Textbooks Are All You Need" style, HumanEval-format), generated with GPT-3.5. License: CC-BY-NC-SA-4.0.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b. URL https://kellerjordan.github.io/posts/muon/.

René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 654–665, 2014. doi: 10.1145/2635868.2635929.

Andrej Karpathy. nanochat: The best chatgpt that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, 2016.

Julia Kopf, Achim Zeileis, and Carolin Strobl. Anchor selection strategies for dif analysis: Review, assessment, and new approaches. *Educational and psychological measurement*, 75(1):22–56, 2015.

Nick Levine, David Duvenaud, and Alec Radford. Introducing talkie: a 13B vintage language model from 1930, April 2026. URL https://talkie-lm.com/introducing-talkie.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984, 2024.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25366–25394, 2025.

Shanchao Liang, Spandan Garg, and Roshanak Zilouchian Moghaddam. The SWE-Bench Illusion: When state-of-the-art LLMs remember instead of reason. *arXiv preprint arXiv:2506.12286*, 2025.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, 2023.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. Student-centered distillation narrows the agentic gap between small and large LLMs. In *International Conference on Machine Learning*, 2026.

Yingwei Ma, Yue Liu, Yue Yu, Yuanliang Zhang, Yu Jiang, Changjian Wang, and Shanshan Li. At which training stage does code data help LLMs reasoning? In *International Conference on Learning Representations*, 2024.

Prasanna Mayilvahanan, Thaddäus Wiedemer, Sayak Mallick, Matthias Bethge, and Wieland Brendel. LLMs on the line: Data determines loss-to-loss scaling laws. In *International Conference on Machine Learning*, 2025.

John Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, 2021.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 94743–94765, 2025.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5191–5198, 2020.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

Ricardo Olmedo, Bernhard Schölkopf, and Moritz Hardt. Computational arbitrage in AI model markets. *arXiv preprint arXiv:2603.22404*, 2026.

OpenAI. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/, August 2024. Published August 13, 2024; updated February 24, 2025.

OpenAI. Why SWE-bench Verified no longer measures frontier coding capabilities. https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, February 2026. Published February 23, 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *International Conference on Machine Learning*, 2025.

Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Sham Kakade, and Surbhi Goel. In good GRACES: Principled teacher selection for knowledge distillation. In *International Conference on Learning Representations*, 2026.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Georg Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10? *arXiv preprint arXiv:1806.00451*, 2018.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, 2019.

Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. Data contamination through the lens of time. *arXiv preprint arXiv:2310.10628*, 2023.

Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. A meta-analysis of overfitting in machine learning. *Advances in neural information processing systems*, 32, 2019.

Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. Observational scaling laws and the predictability of language model performance. In *Advances in Neural Information Processing Systems*, 2024.

Rylan Schaeffer, Noam Levi, Brando Miranda, and Sanmi Koyejo. Pretraining scaling laws for generative evaluations of language models. *arXiv preprint arXiv:2509.24012*, 2025.

Charlie Victor Snell, Eric Wallace, Dan Klein, and Sergey Levine. Predicting emergent capabilities by finetuning. In *First Conference on Language Modeling*, 2024.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince, Sooraj Thomas, et al. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv preprint arXiv:2402.19450*, 2024.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, 2020.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. In *International Conference on Learning Representations*, 2022.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Sang Truong, Yuheng Tu, Rylan Schaeffer, and Sanmi Koyejo. Item response scaling laws: A measurement theory approach for efficient and generalizable neural scaling estimation. *arXiv preprint arXiv:2606.07616*, 2026.

Shiqi Wang, Zheng Li, Haifeng Qian, Chenghao Yang, Zijian Wang, Mingyue Shang, Varun Kumar, Samson Tan, Baishakhi Ray, Parminder Bhatia, et al. Recode: Robustness evaluation of code generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13818–13843, 2023.

You Wang, Michael Pradel, and Zhongxin Liu. Are "solved issues" in SWE-bench really solved correctly? An empirical study. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, 2026.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. In *Advances in Neural Information Processing Systems*, 2025.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. Stronger models are not always stronger teachers for instruction tuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4392–4405, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

John Yang, Kilian Lieret, Carlos Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *Advances in Neural Information Processing Systems*, 38, 2026.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Maxwell Zeff. Inside OpenAI's race to catch up to Claude Code. *WIRED*, March 2026. URL https://www.wired.com/story/openai-codex-race-claude-code/. Accessed: 2026-09-25.

Hansi Zeng, Kai Hui, Honglei Zhuang, Zhen Qin, Zhenrui Yue, Hamed Zamani, and Dana Alon. Can pre-training indicators reliably predict fine-tuning outcomes of LLMs? *arXiv preprint arXiv:2504.12491*, 2025.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *International Conference on Learning Representations*, 2024a.

Chen Zhang, Qiuchi Li, Dawei Song, Zheyu Ye, Yan Gao, and Yao Hu. Towards the law of capacity gap in distilling language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22504–22528, 2025a.

Guanhua Zhang, Ricardo Dominguez-Olmedo, and Moritz Hardt. Train-before-test harmonizes language model rankings. In *International Conference on Learning Representations*, volume 2026, pp. 74776–74808, 2026.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, et al. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37: 46819–46836, 2024b.

Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, et al. SWE-bench Goes Live! In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025b.

A GENERATING TASK DEMONSTRATIONS

We structure our data generation pipeline around two key criteria: to generate training data that is as relevant as possible to SWE-bench, and to generate plenty of it.

We start by synthesizing software engineering tasks at scale. Using SWE-smith (Yang et al., 2026) and Gemma 4 26B-A4B-it (Team et al., 2026), we generate 118.5k tasks out of 1,038 GitHub repositories. This cost roughly $60 under OpenRouter pricing. Adding SWE-smith's 56.5k public instances brings the total to 175k tasks. We ensure that all 14 SWE-bench test repositories, their forks, and clones are strictly held out from task generation.

We then feed the 175k task instances to a lightweight teacher model, collecting the resulting rollouts as training data. We choose Nemotron 3.5 Lightning 30B A3B (Blakeman et al., 2025), with 47% SWE-bench pass@1, due to its high throughput of around 3.3k output tokens/s on a single H100. Running Nemotron with mini-swe-agent v2 (Yang et al., 2024) and a file editing tool, we generate 8 rollouts per task, totalling 1.4 million trajectories and 33.2B tokens. Of these, 15.3B are Nemotron output tokens. The total cost was approximately 1,300 H100 hours.

B TRAINING DETAILS

Our models are nanochat Transformers (Karpathy, 2025). A model of depth ℓ has width $d = 64\ell$, attention heads of dimension 128, ReLU^2 MLPs, parameter-free RMSNorm (including QK-norm), rotary embeddings with base 10^6, untied input and output embeddings, value embeddings on alternating layers, and a logit soft-cap of 15. Every fourth layer attends to the full context; the others use a sliding window of a quarter of the context. We use nanochat's BPE tokenizer with a vocabulary of 32,768 tokens. We count as parameters the weight matrices of the Transformer blocks and the output head, $N = 12\ell d^2 + 32{,}768\, d$, which excludes the input and value embeddings.

Both training phases use Muon (Jordan et al., 2024b) for the block matrices and AdamW for the embeddings and output head, in bf16 with FP8 matrix multiplications and without gradient clipping. Each phase follows a warmup-stable-decay schedule: 40 warmup steps, then a linear decay to 5% of the peak learning rate over the last 65% of steps. Batch size, learning rate, and weight decay scale with the number of training tokens (Table 1). Fine-tuning is a separate run that loads the pre-trained weights with a fresh optimizer state. It packs trajectories into 32,768-token sequences, masks attention across trajectories, and computes the loss on assistant tokens only. Smaller fine-tuning budgets train on prefixes of the same data order, and no sweep run repeats data. We evaluate the final checkpoint of every run.

Table 1: Phase-specific hyperparameters of the sweep, as ranges over runs. Batch size, peak learning rate, and weight decay are set from each run's token budget.

	Pre-training	Fine-tuning
Data	FineWeb-Edu	Lightning trajectories
Sequence length	2,048	32,768
Batch size (tokens)	0.39M to 1.15M	0.26M to 0.72M
Peak Muon learning rate	0.017 to 0.030	0.007 to 0.012
Muon weight decay	0.48 to 0.05	0.35 to 0.04

The 192-hour speedrun uses the same recipe at depth 32 ($N = 1.68$B), with 6.7B pre-training tokens and a fine-tuning batch of 1.3M tokens. Its fine-tuning schedule follows the fraction of the time budget spent rather than the step count. In total, it used 191.7 B200 hours (17.8 for pre-training, 173.9 for fine-tuning).

C PREDICTABLE PERFORMANCE

Our sweep shows that fine-tuned pass@1 is predictable for models trained from scratch (Section 2.1). Here we ask whether the same holds for a heavily pre-trained model: can we predict how well a late pre-training checkpoint fine-tunes from the fine-tunes of its early checkpoints alone?

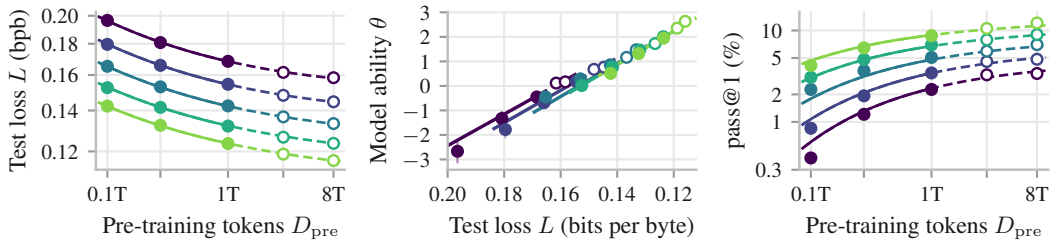


Figure 9: Scaling laws for fine-tuning SmolLM3 checkpoints on D_{sft} = **0.18B** (●), **0.42B** (●), **1B** (●), **2.37B** (●), or **5.6B** (●) tokens, fitted on the checkpoints up to 1T pre-training tokens (filled) and extrapolated to 3T and 8T (hollow, dashed). *Left*: Test loss follows Equation 7. *Middle*: Model ability follows Equation 6 with the discount fitted in Section 4, one line per SFT budget. *Right*: Chaining both laws predicts pass@1 from D_{pre} and D_{sft} alone, to within 0.25pp on average on the held-out checkpoints.

We take five SmolLM3 3B checkpoints, with D_{pre}={0.1, 0.3, 1, 3, 8}T tokens, and fine-tune each on D_{sft}={0.18, 0.42, 1.0, 2.4, 5.6}B tokens of mini-coder-trajs. We evaluate both test loss L on teacher trajectories and pass@1 on SWE-bench. Fitting scaling laws on $D_{\text{pre}} \leqslant$ 1T, we aim to extrapolate loss and pass@1 to D_{pre}={3, 8}T at each of the SFT budgets.

We take the scaling laws developed in Section 2.1, and adapt them to this new highly overtrained setting, where D_{pre} exceeds D_{sft} by up to four orders of magnitude. We start by modeling test loss. Whereas in Section 2.1 D_{pre} only modulated the amplitude of the D_{sft} term, we find that in this highly overtrained setting, pre-training also changes the SFT scaling exponent:

$$L = E + B\left(1 + AD_{\text{pre}}^{-\alpha}\right) D_{\text{sft}}^{-\beta - c \ln D_{\text{pre}}}. \tag{7}$$

Fitted on the checkpoints up to 1T ($E = 0.078$, $B = 0.047$, $A = 0.39$, $\alpha = 0.35$, $\beta = 0.20$, $c = 0.009$), the law predicts the test loss of the held-out checkpoints to within 0.33% (Figure 9 left). Pre-training steepens the SFT exponent β from 0.18 at 0.1T to 0.22 at 8T tokens.

For pass@1, we fit the Rasch model of Equation 2 as before, and relate ability to test loss with the law of Section 4 (Equation 6). We keep its discount at the values fitted there ($r = 0.10$, $D_0 = 0.35$B) and fit only θ_0 and κ, by weighted least squares on the checkpoints up to 1T ($\kappa = 68.1$).[9] Chaining Equations 7 and 6, we predict the pass@1 of the held-out checkpoints to within 0.25pp on average (4% relative error; Figure 9, right). Across 0.1T to 8T tokens of pre-training, the checkpoints thus behave as one student in the sense of Section 4.

D LIGHTNING FINE-TUNES

We repeat the analysis of Section 4 with our own Lightning trajectories (Figure 10). These trajectories exist as two SFT datasets, processed differently for different experiments. The *stripped* dataset, which our speedruns train on, merges the SWE-smith and SWE-smith-v3 trajectories, keeps only the issue text and the agent's turns, and allows one tool call per turn. We fine-tune SmolLM3 base on it with 0.2B to 5B tokens, alongside a 3.7B-parameter speedrun, pre-trained on 24B web tokens and fine-tuned with 0.5B to 7B tokens, and our 192-hour speedrun. The *tool-call* dataset keeps the SWE-smith trajectories in their original chat format, with a system prompt that lists the tools. We fine-tune SmolLM3 mid-train and Qwen 3 4B on it with 0.2B tokens up to one epoch (4.8B tokens). Each dataset is evaluated on SWE-bench Verified with its own harness, matched to its format.

Why we fit the two datasets separately. We measure test loss on the same held-out Lightning trajectories for every model, rendered in each model's training format. In both formats, we score the same turns and only content tokens, that is, reasoning and tool calls. Scoring the chat template's tool-call tags as well, which are nearly free to predict, would lower the tool-call models' loss by 3%.

[9] Without the discount, a single ability line under-predicts all ten held-out fine-tunes, by 0.6pp on average. Section 4 fits r and D_0 on data that include the 8T checkpoint; refitting them without it ($r = 0.09$, $D_0 = 0.37$B) leaves the held-out error unchanged.

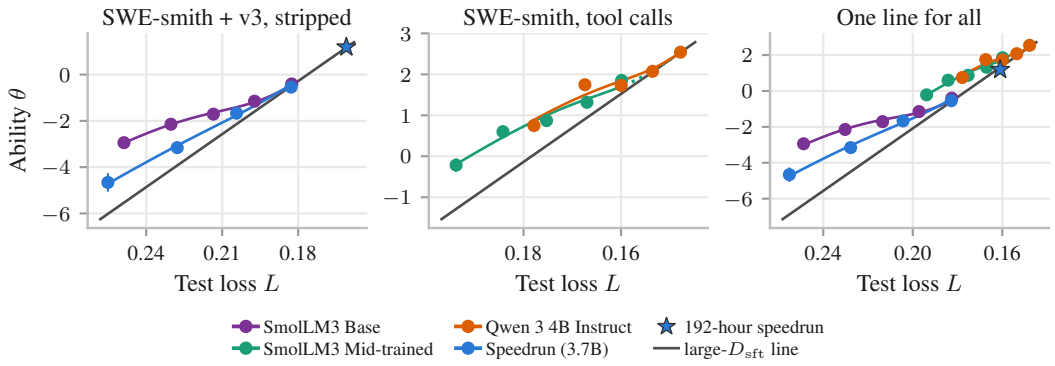


Figure 10: Ability against test loss for models fine-tuned on Lightning trajectories, in two differently processed SFT datasets. *Left*: SmolLM3 base and a 3.7B speedrun on the stripped dataset, with our 192-hour speedrun. *Middle*: SmolLM3 mid-train and Qwen 3 4B on the tool-call dataset (dotted: SmolLM3 mid-train extrapolated along its loss law). *Right*: one line for all models. Curves follow Equation 6 with r and D_0 fitted per model, and approach the black line for large D_{sft}.

Even so, the two losses are not on a common scale: each format predicts the same tokens in a different context, and each dataset's pass@1 comes from a different harness. Loss and ability therefore need not relate in the same way across the two datasets, just as they differ between the two teachers of Section 4, so we fit Equation 6 separately for each dataset. For completeness, a single line for all models fits almost as well ($p = 0.50$; Figure 10, right), but it compares losses measured in different formats, and we do not interpret it.

Consistency with Section 4. Within each dataset, models with very different pre-training share one ability curve in test loss. On the stripped dataset, SmolLM3 base, pre-trained on 8T tokens of web and code, and the 3.7B speedrun, pre-trained on 24B web tokens, approach the same line, $\theta = 13.5 - 76.4\,L$. At their largest fine-tuning budgets, both sit within 0.08 of this line. Chaining the loss and ability laws reproduces the pass@1 of all 10 fine-tunes on this dataset with a root-mean-square error of 0.22pp, including our 192-hour speedrun (25.5% measured, 25.3% predicted). On the tool-call dataset, SmolLM3 mid-train and Qwen 3 4B approach $\theta = 14.8 - 83.1\,L$, with a root-mean-square error of 1.4pp over 10 fine-tunes; the largest miss, 2.9pp, is Qwen 3 4B at 0.44B tokens. As in Section 4, models start above the line early in fine-tuning, by a similar factor: r ranges from 0.07 to 0.16, against 0.10 in Section 4.

Differences. The early discount fades more slowly here: D_0 ranges from 0.9B to 2.8B tokens, against 0.35B in Section 4. It also varies between models, so we fit r and D_0 per model; sharing them across models, as in Section 4, raises χ^2 from 32 to 116 with six fewer parameters. These results are preliminary: each dataset has only two pre-trained models, and the two datasets differ in both processing and evaluation.

E Additional test sets

We build four test sets by crossing two sources of bugs with two groups of codebases (Table 2). Each instance follows the SWE-bench format: an issue, a containerized repository, a gold patch, and the tests used for grading. Tests in `FAIL_TO_PASS` fail before the gold patch and pass after it; tests in `PASS_TO_PASS` pass in both states. We keep an instance only if its gold patch passes this check in the instance's own container.

Codebases. The SWE-bench codebases are 23 repositories from the test, dev, and train splits of SWE-bench (Jimenez et al., 2024); for PR-based bugs, the twelve Verified repositories alone yield too few recent fixes. The other codebases are 28 popular, actively maintained Python repositories tested with pytest. They appear in no SWE-bench split and in none of our training data, where we match repository names case-insensitively and also exclude renamed repositories and forks.

Table 2: Our four held-out test sets.

Bugs	Codebases	Repositories	Instances
Synthetic	SWE-bench	9	400
Synthetic	Other	12	248
PR-based	SWE-bench	22	367
PR-based	Other	24	401

Synthetic bugs. We prompt Opus 4.8 to inject a small semantic bug into a single function of a test-covered source file, changing at most eight lines and no comments, docstrings, or imports. The gold patch reverts the bug, and the test patch is empty. We run the existing tests with and without the bug: tests that the bug breaks form `FAIL_TO_PASS`, and tests that pass in both states form `PASS_TO_PASS`. Opus 4.8 then writes an issue that reports the symptom without prescribing the fix, and we drop issues that quote the fix. In SWE-bench codebases, we inject bugs only into files outside the footprint of SWE-bench Verified: no Verified gold patch edits them, and no Verified test patch edits the tests that cover them. These instances run in the Verified containers; for the other codebases, we build environments with SWE-smith (Yang et al., 2026).

PR-based bugs. We mine pull requests merged between January and August 2026 that change tests and a single non-test Python file, in at most two hunks and ten lines. Opus 4.8 keeps those that fix a bug solvable from the text of the pull request or its issue. Each instance starts from the pull request's base commit, with its source change as the gold patch and its test change as the test patch. We grade on the tests in the pull request's test files. The issue is the linked GitHub issue, used verbatim for about a third of instances; where there is none, or where it quotes the gold code, Opus 4.8 rewrites the pull request description or issue into a report of the symptom. We build all environments with SWE-smith, with an Opus 4.8 agent repairing the installation where automatic setup fails.

F Difficulty shift: training details

From-scratch models. The 21 from-scratch models use the architecture and optimizer of Appendix B, but train only on agentic trajectories, from random initialization. They span depths 16 to 42 (N = 235M to 3.7B) and 1.9B to 24B training tokens; the 18 models up to depth 24 cover two data recipes with three data seeds each. Besides mini-coder-trajs-400k, their data includes NVIDIA's SWE-Zero trajectories. Ten models train on a fixed 56:44 mix of the two, and the other eleven first train on SWE-smith, SWE-Zero, and SWE-Hero trajectories before finishing on the same mix.

SmolLM3 fine-tunes. We fine-tune 14 checkpoints of SmolLM3 3B (Bakouch et al., 2025): 13 from its three pre-training stages, with 0.1T to 11.2T tokens, and its mid-trained checkpoint. We fine-tune each with five budgets, from 0.18B tokens up to one epoch of mini-coder-trajs-400k,[10] and use 69 of the 70 fine-tunes. We use AdamW ($\beta = (0.9, 0.95)$, weight decay 0.1) with a peak learning rate of 2×10^{-5}, 3% warmup, and cosine decay to 10%, clip gradients at 1.0, and pack batches of 524,288 tokens into 32,768-token sequences, with the loss on assistant tokens only.

Evaluation. We evaluate all models with mini-swe-agent (Yang et al., 2024) at temperature 0.7, with 10 attempts per problem on the 446 SWE-bench Verified problems evaluated for both cohorts. For the leaderboard comparison of Appendix H, the strongest from-scratch model gets 40 to 80 more attempts on 103 problems it rarely or never solves, for up to 90 in total. Each cohort uses the scaffold settings of its training data: the from-scratch models run without a system prompt and for at most 100 steps of 2,048 tokens, the SmolLM3 fine-tunes with mini-coder's system prompt and no step limit.

G Generalization: training details

Table 3 lists the twelve models of Sections 3.2 and 3.3. The six speedruns follow Appendix B and fine-tune on the stripped Lightning dataset of Appendix D; all but one first pre-train on FineWeb-Edu.

[10] This experiment was performed prior to generating our Lightning trajectories.

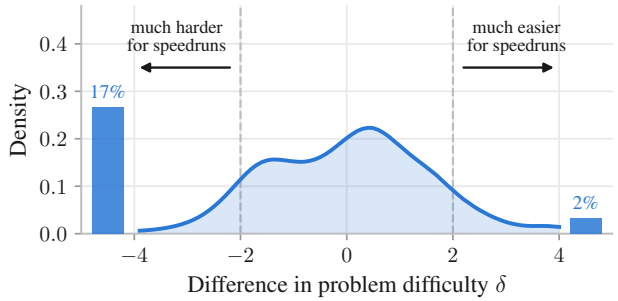


Figure 11: Per-problem difficulty differences δ between our speedruns and the SWE-bench leaderboard at matched model ability. The bars collect the mass beyond the plotted range. Compared to the leaderboard, our speedruns find 35% of problems much harder: at the same pass@1, they solve a narrower subset of problems more reliably.

The SmolLM3 fine-tunes use the recipe of Appendix F. Those starting from SmolLM3 base train on the same stripped dataset as the speedruns; those starting from the mid-trained checkpoint train on the tool-call dataset, which holds only the SWE-smith part of our Lightning trajectories. Each is evaluated in the harness matching its training format, with 5 attempts per problem on SWE-bench Verified and the four new test sets (1 on Verified for the two depth-20 speedruns and the first depth-24 speedrun).[11]

Table 3: The twelve models of Sections 3.2 and 3.3.

Model	N	Pre-training tokens	Lightning SFT tokens
Speedrun, depth 20	435M	3.5B	3.5B, 7.0B
Speedrun, depth 24	730M	5.5B	5.5B
Speedrun, depth 24	730M	–	23.4B
Speedrun, depth 28	1.14B	4.6B	21.5B
Speedrun, depth 32 (192 hours)	1.68B	6.7B	37.8B
SmolLM3 3B base	3B	8T	0.2B, 1.0B, 2.2B
SmolLM3 3B mid-trained	3B	11T	0.2B, 0.44B, 2.1B

Elicitation. For Section 3.3, we fine-tune each of the twelve models for three epochs on the 230M-token elicitation dataset,[12] starting from its Lightning fine-tune. The speedruns use batches of 2^{20} tokens and a Muon learning rate of 0.023 without weight decay, held for the first half of training and decayed linearly to zero over the second. The SmolLM3 fine-tunes use the recipe of Appendix F, with each model's own chat template.

H Difficulty differences against the SWE-bench leaderboard

We compare our speedruns, the 21 from-scratch models of Section 3.1, against the public SWE-bench leaderboard using the Rasch model with differential item functioning of Section 3. The leaderboard covers a wide ability range, from Claude 2 and GPT-3.5 to the agentic models of 2026. We plot the difficulty differences δ between our speedruns and the leaderboard in Figure 11. We find high agreement for most problems: 58% of the mass falls within $(-2, 2)$. The tail, however, is heavy and one-sided: the leaderboard models find 35% of problems much easier compared to our speedruns. Thus, at the same pass@1, our speedruns solve a narrower subset of problems, but more reliably.

Our speedruns thus have substantial gaps relative to the broader set of models on the SWE-bench leaderboard. We speculate that this is not a fault of specialization, which Olmedo et al. (2026)

[11] Some curves are visibly concave, but adding a quadratic term to Eq. 5 changes γ by less than 0.1 points.

[12] Specifically, 100M tokens of HumanEval-like docstring-to-Python exercises (Jina AI, 2023), 100M tokens of mathematical reasoning from MetaMathQA-GSM (Yu et al., 2023) and Orca-Math (Mitra et al., 2024), and 30M tokens of question answering from the MMLU auxiliary train set.

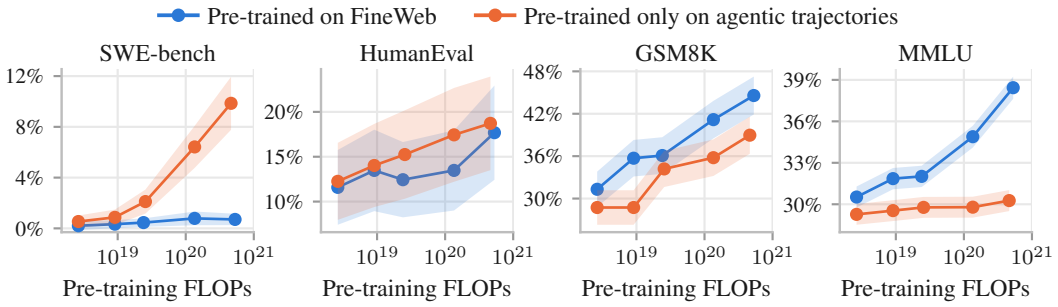


Figure 12: Downstream performance after identical SFT on a shared elicitation dataset, as a function of base-training compute. From left to right: SWE-bench, HumanEval, GSM8K, and MMLU. Scaling trajectory training yields large gains on SWE-bench, similar HumanEval performance to web training, and improving but lower GSM8K accuracy; on MMLU, the speedruns improve little while the web models improve substantially. Shaded bands show approximate 95% confidence intervals reflecting evaluation uncertainty.

find to monotonically improve pass@100, but rather that the weaker leaderboard models are under-elicited for SWE-bench-like tasks. Such under-elicited models can be surprisingly unreliable even on straightforward issues, yet still occasionally solve harder ones. Our speedruns show the opposite profile: they reliably execute simple fixes but fail outright once a problem is difficult enough.

I From-scratch scaling on web vs agentic trajectories

We elicit downstream capabilities to test how far the benefits of training on agentic trajectories extend beyond agentic bug fixing. We compare two from-scratch scaling sweeps: *speedrun* models, whose base training uses only agentic trajectories, and *web* models, whose base training uses FineWeb-Edu (Lozhkov et al., 2024), a high-quality web corpus. We train both families with nanochat (Karpathy, 2025) at five model sizes and compare downstream performance as a function of base-training compute.

We evaluate SWE-bench alongside three other benchmarks: HumanEval, GSM8K, and MMLU. To elicit these capabilities, we curate a shared 330M-token dataset[13] and fine-tune every model for three epochs using the same SFT protocol. All four benchmarks are evaluated after this joint fine-tuning.

Figure 12 shows that the benefits of scaling depend strongly on the base training data. On SWE-bench, speedrun performance rises from 0.5% to 9.9% across the sweep, while the web models remain below 1%. On HumanEval, both families improve overall and achieve similar performance, although uncertainty is substantial. The benefits of trajectory training extend to mathematical reasoning: speedrun accuracy on GSM8K rises from 29% to 39%, but remains below that of the web models, which improve from 31% to 45%. On MMLU, however, the speedruns improve little, from 29% to 30%, while the web models improve from 31% to 38%.

Training on agentic trajectories thus supports downstream code generation and mathematical reasoning after elicitation, but its benefits transfer unevenly across tasks. Web pre-training provides broader gains, while trajectory training retains a large advantage on SWE-bench.

J Additional related work

New test sets and contamination. Collecting a new test set with the original procedure tests for overfitting directly. On CIFAR-10 and ImageNet, accuracy drops on the new test sets, but gains on the original test sets carry over (Recht et al., 2018; 2019), and Kaggle competitions show little

[13] Specifically, 100M tokens of agentic coding trajectories, 100M tokens of HumanEval-like docstring-to-Python exercises (Jina AI, 2023), 100M tokens of mathematical reasoning from MetaMathQA-GSM (Yu et al., 2023) and Orca-Math (Mitra et al., 2024), and 30M tokens of question answering from the MMLU auxiliary train set.

overfitting from reusing test sets (Roelofs et al., 2019). A new version of GSM8K, in contrast, reveals overfitting in some model families (Zhang et al., 2024b). Perturbing existing problems is a cheaper alternative, for math (Mirzadeh et al., 2025; Li et al., 2024; Srivastava et al., 2024) and for code (Wang et al., 2023); our Rephrased set follows ReCode in renaming identifiers. Test data created after a model's training cutoff avoids contamination (Roberts et al., 2023; Jain et al., 2025a). SWE-bench has specific weaknesses: issues can reveal their solutions and tests can be weak (Aleithan et al., 2024; Wang et al., 2026), and frontier models name the buggy file from the issue alone far more often than on other repositories, a sign of memorization (Liang et al., 2025). Benchmarks built from fresh issues address the latter (Zhang et al., 2025b; Badertdinov et al., 2026), as do our pull-request bugs. Our synthetic bugs follow mutation testing, which injects small faults into programs to assess test suites (DeMillo et al., 1978; Jia & Harman, 2011); test suites that catch such faults also tend to catch real ones (Just et al., 2014). Comparing cohorts on a new test set at matched Verified score, as in Equation 5, measures what the robustness literature calls effective robustness (Taori et al., 2020; Miller et al., 2021).

Specialization and jagged capabilities. Specializing a small model on one task can trade away general ability (Fu et al., 2023). Some skills do spread: code in the pre-training mix improves reasoning well beyond code (Aryabumi et al., 2025; Ma et al., 2024), and reinforcement learning on software engineering data improves several out-of-domain tasks (Wei et al., 2025). Our speedruns show both patterns: training on agentic trajectories improves HumanEval and GSM8K, but not MMLU. Across pre-trained language models, benchmark scores are well explained by a few latent factors (Burnell et al., 2023; Ruan et al., 2024), and after identical fine-tuning by essentially one (Zhang et al., 2026). Our speedruns break this pattern: at matched SWE-bench score, they trail SmolLM3 fine-tunes by 24 to 29 points on MMLU, a sharp case of what Dell'Acqua et al. (2026) call a jagged frontier.

Does loss capture ability? Predicting performance through loss presumes that models with equal loss perform alike, as Du et al. (2024) find for pre-training loss on a fixed corpus. But equal pre-training loss can hide different downstream performance (Liu et al., 2023; Tay et al., 2022), and loss can be a poor guide to task quality (Isik et al., 2025) and to performance after fine-tuning (Zeng et al., 2025). How losses on different distributions relate also depends on the pre-training data (Brandfonbrener et al., 2025; Mayilvahanan et al., 2025). Closer to our setting, the likelihood of reference solutions gives the most stable scaling law for pass@k (Schaeffer et al., 2025), and perplexity and downstream rankings agree after identical fine-tuning (Zhang et al., 2026). The one-student hypothesis concerns the loss on held-out specialization data after fine-tuning, compared across models with very different pre-training. It has a parallel in robustness, where pre-trained models show effective robustness early in fine-tuning that vanishes by convergence (Andreassen et al., 2022), much as the discount in Equation 6 fades within the first billion fine-tuning tokens.

Teachers and students. Knowledge distillation trains a student on a teacher's outputs (Hinton et al., 2015); training on sequences the teacher generates, as we do, is its sequence-level form (Kim & Rush, 2016), now a standard way to pass reasoning and agentic behavior to small models (Mukherjee et al., 2023; Guo et al., 2025; Lyu et al., 2026). Better teachers do not always make better students: larger teachers can yield worse students (Cho & Hariharan, 2019; Mirzadeh et al., 2020), the best teacher's size grows with the student's (Zhang et al., 2025a), and distillation scaling laws predict that student loss eventually worsens as the teacher improves (Busbridge et al., 2025). The same holds for fine-tuning on generated data: stronger response generators are not necessarily better teachers (Xu et al., 2025), QwQ-32B beats the stronger DeepSeek-R1 (Guha et al., 2025), and small students learn better from simpler reasoning traces (Li et al., 2025). Panigrahi et al. (2026) predict which teacher suits a student from the student's own gradients. These works tie the choice of teacher to the student's size or to how compatible teacher and student are. In our setting, students of similar size prefer different teachers depending on how far along they are, and imitation loss predicts where the preference flips.