Data Quality Doesn't Wash Out

Benefits from Improved Pretraining Data Persist Across Scale and Post-Training

DatologyAI Team[1]

Post-training receives much of the credit for the capabilities of modern language models, yet it largely elicits what pretraining already provides. We ask how much better pretraining data improves a language model, and whether that improvement survives the stages of training that follow. We build two pretraining mixtures from the same 39 public datasets: a strong hand-designed baseline and a mixture curated with Datology Curation Studio. We then train pairs of mixture-of-experts models on each, from 276M to 30.4B total parameters, with matched recipes for everything except the pretraining data through the steps of pretraining, context extension, supervised fine-tuning (SFT), and reinforcement learning with verifiable rewards (RLVR). To make controlled comparisons like this workable across datasets and scales, we use a training recipe that removes two hyperparameters by design and transfers most of the rest from small-scale tuning runs to the largest models without further sweeps. Across all five scales studied, models trained on curated data achieve 11–12% lower bits per byte loss on long-form benchmark evaluations than their baseline counterparts. These gains from better pretraining data persist through post-training. After RLVR, the curated 30B-A3B model achieves an average benchmark score of 46.8 across 25 benchmarks, compared with 37.7 for its baseline counterpart. The curated-data scaling trend matches the final performance of the 30B-A3B baseline model, which is pretrained on one trillion tokens, at 6× less pretraining compute.

1 Introduction

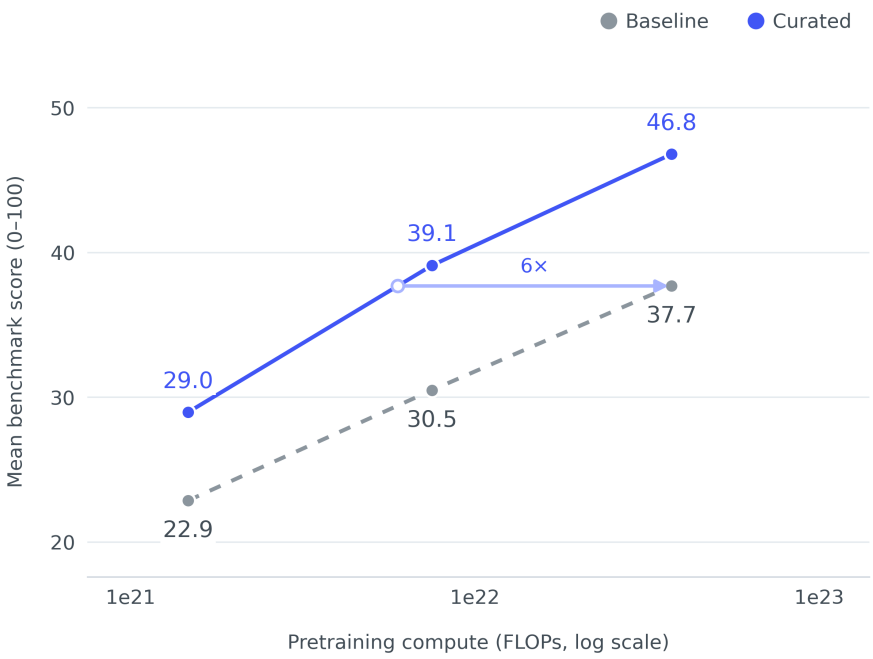


Figure 1 Average score over 25 benchmarks for post-trained models at three sizes, plotted against the compute used to pretrain them. Following the CURATED trend, a CURATED model would match the largest BASELINE model's score with about 6× less pretraining compute.

[1]See Contributions and Acknowledgments (§ 8) for full author list.

What practitioners ultimately care about are specific capabilities of final, post-trained models, often run inside specific harnesses. While much attention goes to the harnesses and post-training recipes that make these capabilities emerge, pretraining plays a pivotal role in making that emergence both possible and easier to elicit (Foster et al., 2025; Chen et al., 2025a; Zhang et al., 2025). A surprisingly high-fidelity mental model of RL is that it sharpens the next-token distribution, converting pass@k performance into pass@1 performance (Yue et al., 2025), and earlier post-training stages show the same pattern (Malladi et al., 2026; Ahmed, 2026). This suggests that coverage of the capabilities of interest is the key objective of pretraining.

Data curation acts directly on that coverage: it can align the pretraining distribution with capabilities of interest and remove patterns antithetical to downstream goals, such as corrupted or off-topic data. We hypothesize that curating the pretraining data alone can have a dramatic effect on final-model quality, one that holds across scale and survives post-training.

To test this hypothesis, we evaluate Datology Curation Studio, a product that integrates large-scale redundancy control, quality selection, synthetic data generation, domain- and language-aware processing, and learned dataset composition into a single system for turning raw data into balanced datasets ready for training. Our goal is to evaluate the aggregate impact of this system, including interactions among its components, because together these technologies represent the state of the art in preparing high-quality pretraining data.

Isolating the effect of data curation requires more control than comparing independently developed models, where the effects of data may be entangled with changes in architecture, tokenizer, optimization, training budget, or post-training. We therefore construct a carefully controlled paired intervention from the same collection of 39 publicly curated source datasets: we assemble the BASELINE pretraining mixture by hand and build the CURATED pretraining mixture by applying Datology Curation Studio to filter, transform, augment, and compose those inputs. We then match the surrounding model-development pipeline through pretraining and post-training.

1.1 Research questions

We organize the study around three questions:

1. **Leverage across scale.** At matched architecture, training budget, and training protocol, how large is the effect of data curation, and how does the effect change as training is scaled up?
2. **Persistence through post-training.** Do context extension, supervised fine-tuning (SFT), and reinforcement learning with verifiable rewards (RLVR) consume or enhance the advantage created by pretraining on well-curated data?
3. **The nature of gains.** Where do the gains of curation appear across capabilities? How consistent are the quality gaps between models trained on CURATED and BASELINE data across general knowledge, analytical reasoning, mathematics, code, multilingual ability, instruction following, and tool use?

1.2 Findings at a glance

Across five training scales, the CURATED models achieve 11–12% lower bits per byte (BPB) than their matched BASELINE counterparts, and the relative gain is essentially unchanged from the smallest scale to the largest.

The advantage persists through matched context extension and post-training. Over 25 benchmarks, the post-trained 30B-A3B CURATED model achieves an average score of 46.8, compared with 37.7 for its BASELINE counterpart, with gains spanning general reasoning, mathematics, coding, multilingual ability, and tool use. The 12B-A1.4B CURATED model, pretrained on 44% as many tokens, scores higher than the 30B-A3B BASELINE model on the 25-benchmark mean (39.1 versus 37.7) and in mathematics and code, despite using fewer than half the active parameters and roughly a fifth of the pretraining compute. Following the CURATED scaling

trend, a CURATED model would match the 30B-A3B BASELINE model's score with about 6× less pretraining compute (Figure 1).

2 Data and Baselines

Our BASELINE and CURATED models differ only in their pretraining data, so the gap between them measures what curation adds over the BASELINE. That gap is meaningful only if the BASELINE is itself strong, so we first compare it against simpler public-data alternatives and then describe the CURATED treatment.

2.1 Source collection

Both training mixtures draw on the same 39 publicly curated source datasets. The collection includes recent Nemotron pretraining and synthetic SFT-style releases, The Stack v3, FineWeb, and DCLM (NVIDIA, 2025; Penedo et al., 2024; Li et al., 2024a), together with other web, mathematical, code, academic, and specialized sources. It covers the major capability areas of a general-purpose language model rather than favoring a narrow downstream task.

Both the BASELINE and CURATED mixtures were designed at a scale of 10 trillion tokens. To reduce dataset-construction compute for this study, we materialized 2 trillion tokens of the BASELINE mixture and 5 trillion tokens of the CURATED mixture. Every training run in this report fits within these token volumes, as the largest models are pretrained for one trillion tokens and all other models for fewer.

2.2 Strong baseline

We construct the BASELINE as a hand-designed mixture of the 39 sources. Constructing it required substantial manual effort: we relied on expert judgment, patterns from recent OLMo 3, Nemotron 3 Nano, and IBM Granite 4.1 recipes (Team Olmo et al., 2025; NVIDIA, 2025; IBM Research, 2026), and small-scale training experiments to choose source allocations. The goal is to approximate the strongest mixture achievable from these public datasets without sophisticated curation tooling.

To test whether the BASELINE is a strong comparison point, we compare it against five alternative data configurations. We train a 4B-A0.6B model for 190 billion tokens on each configuration using the standard training protocol in § 3.

The alternatives are:

- **FineWeb-Edu**: web text filtered for educational quality (Penedo et al., 2024).
- **peS2o**: cleaned and filtered open-access scientific papers (Soldaini and Lo, 2023).
- **Web/Math/Code A** (mixture A): FineWeb-Edu, MegaMath-Web (Zhou et al., 2025), and The Stack v2 (Lozhkov et al., 2024) in a 2:1:1 web:math:code mixture.
- **Web/Math/Code B** (mixture B): Nemotron-CC v2 (NVIDIA et al., 2025), Nemotron-CC-Math v1 (Mahabadi et al., 2025), and The Stack v3 (Lozhkov et al., 2026), also in a 2:1:1 web:math:code mixture.
- **Web/Math/Code B (1:1:1)**: an equal-proportion variant of Web/Math/Code B.

Measuring predictive quality. At small scales, task accuracy is noisy, often near chance, and depends heavily on whether the model has learned each benchmark's answer format. We therefore evaluate these checkpoints, and the models throughout this report, with bits per byte (BPB) on reference answers: the negative log-likelihood of the evaluated text, measured in bits and normalized by its byte length.

$$\mathrm{BPB} = -\frac{1}{N_{\text{bytes}}} \sum_{t \in \mathcal{S}} \log_2 p_\theta(x_t \mid x_{<t}), \tag{2.1}$$

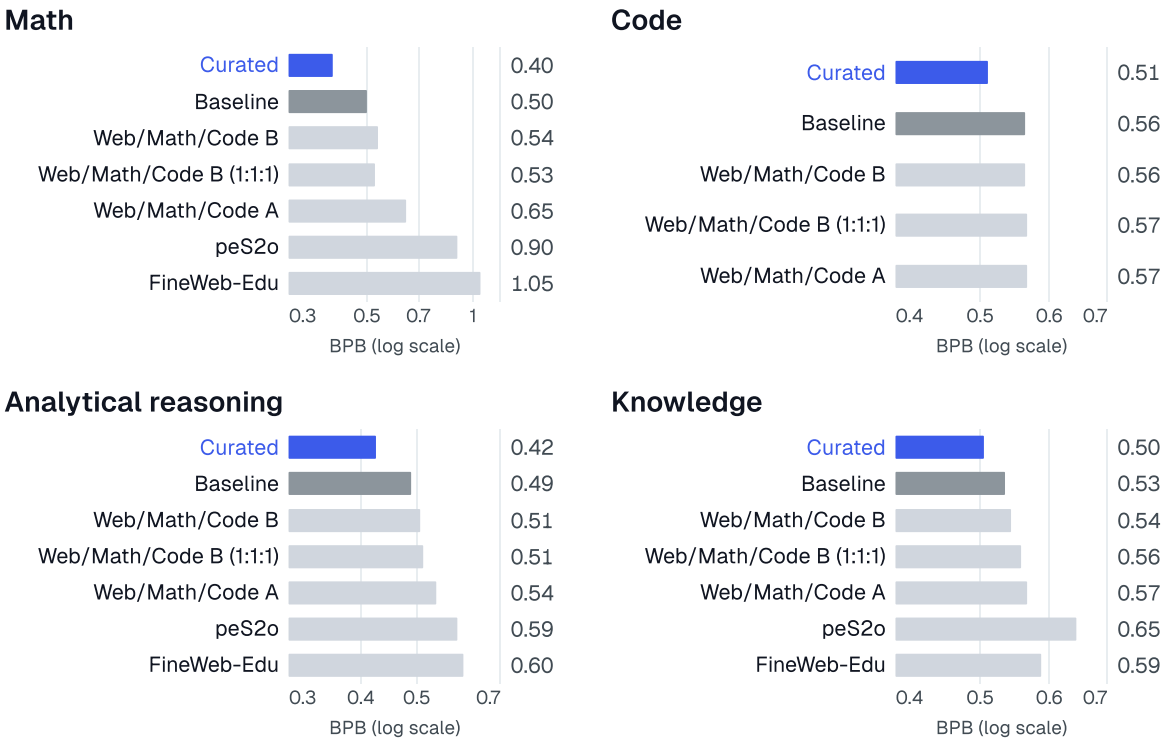


Figure 2 The BASELINE mixture matches or beats every public alternative in every category, and the CURATED mixture improves on it further. Bits per byte (BPB, lower is better; log scale) for 4B-A0.6B models trained for 190B tokens on each data configuration. The Code panel omits the single-domain controls (peS2o, FineWeb-Edu), whose code BPB is far higher (Table 1).

where $\mathcal{S}$ indexes the tokens of the evaluated text and N_{bytes} is its length in bytes.

Byte normalization provides a common unit of measurement across tokenizers. Lower BPB indicates better prediction of the reference answer. We group the BPB tasks into four categories:

- **Analytical reasoning:** scientific and logical reasoning, including GPQA, DROP, science and medicine subjects from MMLU, and selected BBH tasks involving logical deduction, spatial reasoning, and tracking objects.
- **Knowledge:** factual, commonsense, and language understanding, including Natural Questions, ARC-Challenge, Social IQA, and selected MMLU and BBH tasks covering subjects such as history, law, economics, and language understanding.
- **Mathematics:** word problems, competition mathematics, and quantitative reasoning, including GSM8K, MATH-500, AMC23, the SAT Math and AQuA-RAT subsets of AGIEval, mathematics subjects from MMLU, and BBH multistep arithmetic.
- **Code:** program generation and execution reasoning, including HumanEval+, MBPP+, BigCodeBench, CRUXEval input and output prediction, and multilingual MBPP tasks.

These tasks also include our released freeform question-answering datasets, which extend standard benchmarks with verified reference answers suitable for likelihood evaluation.[2] We average task-level BPB within each category and report overall BPB as the mean of the four category scores.

[2] https://huggingface.co/collections/DatologyAI/benchmark-freeform-rewrites

Training data	Code	Math	Analytical	Knowledge	Overall
CURATED	**0.510**	**0.400**	**0.425**	**0.504**	**0.460**
BASELINE	<u>0.563</u>	<u>0.499</u>	<u>0.489</u>	<u>0.534</u>	<u>0.521</u>
Web/Math/Code B	<u>0.563</u>	0.537	0.507	0.543	0.538
Web/Math/Code B (1:1:1)	0.566	0.528	0.512	0.557	0.541
Web/Math/Code A	0.566	0.647	0.542	0.567	0.580
peS2o	1.808	0.903	0.589	0.645	0.986
FineWeb-Edu	1.825	1.049	0.601	0.588	1.016

Table 1 BPB comparison across training-data configurations. Web/Math/Code A combines FineWeb-Edu, MegaMath-Web, and The Stack v2. Web/Math/Code B combines Nemotron-CC v2, Nemotron-CC-Math v1, and The Stack v3. Both use a 2:1:1 web:math:code ratio. The equal-proportion mixture is shown separately. Overall is the unweighted mean of the four domain scores. Lower is better; bold marks the best value in each column, and underlining marks the second best.

Figure 2 shows the category-level comparison, and Table 1 reports the exact values. FineWeb-Edu and peS2o are single-domain controls, not general-purpose mixtures, and contain almost no code, so we omit them from the code panel of Figure 2; Table 1 still reports their code BPB (1.825 and 1.808, respectively), far worse than that of any mixture.

Our BASELINE mixture reaches an overall BPB score of 0.521, 3.0% lower than that of mixture B, the strongest alternative we studied, and approximately 10% lower than the score for mixture A. The single-domain controls perform substantially worse.

The alternative mixes also illustrate which interventions matter. Comparing Web/Math/Code A (whose web component is FineWeb-Edu) against FineWeb-Edu alone shows that blending specialized mathematics and code data into web text lowers BPB in every category. The newer data sources used in mixture B lower overall BPB from 0.580 to 0.538, a reduction of about 7%. The largest domain-level improvement from the A-to-B shift is in mathematics, where BPB falls from 0.647 to 0.537, while BPB on code changes only slightly, from 0.566 to 0.563. The equal-proportion variant of mixture B reaches an overall BPB of 0.541, close to the 0.538 of the original mixture B.

Mixture B also gets a free pass on scale: unlike the BASELINE mixture, it cannot reach 10 trillion tokens. Its 2:1:1 allocation calls for 2.5 trillion mathematics tokens, but Nemotron-CC-Math-v1 contains only about 133 billion tokens (Mahabadi et al., 2025), so filling the budget would take about 19 passes over it. Muennighoff et al. (2023) find negligible changes in loss or downstream performance through approximately four repetitions, after which downstream performance begins to decline. The BASELINE mixture, by contrast, reaches a 10-trillion-token budget while reusing no source more than approximately four times.

Curation produces a clear gain on top of this strong baseline, reducing overall BPB to 0.460. This is an 11.8% improvement over the BASELINE mixture and a 14.4% improvement over mixture B, and BPB scores improve in every evaluation domain.

2.3 The Datology Curation Studio treatment

We build the CURATED mixture by applying Datology Curation Studio to the same raw source collection. The Curation Studio decides which examples to keep, how to expand scarce high-value data, and how to weight sources in the final mixture. We describe the main operations below, though implementation details are outside the scope of this report.

For web and multilingual data, the Curation Studio combines model-based filtering, embedding-based selection, and language-aware curation, drawing upon web-corpus curation pipelines (Wenzek et al., 2020; Rae et al., 2021; Penedo et al., 2023; Soldaini et al., 2024; Penedo et al., 2024; Li et al., 2024a; Su et al., 2025), learned

quality and importance scoring (Xie et al., 2023b; Wettig et al., 2024), and domain taxonomies for web text (Wettig et al., 2025). Luxical models (DatologyAI Team, 2025b) produce high-throughput embeddings for classification and corpus organization, and our ÜberWeb study describes how we adapt filtering, embedding, synthesis, and mixing across languages (DatologyAI Team, 2026), extending earlier multilingual corpora (Xue et al., 2021; Laurençon et al., 2022; Nguyen et al., 2024; Penedo et al., 2025). Synthetic generation extends the source-rephrasing approach of WRAP (Maini et al., 2024) and our BeyondWeb research (DatologyAI Team, 2025a), within a broader line of work on synthetic pretraining data (Gunasekar et al., 2023; Li et al., 2023b; Abdin et al., 2024; Ben Allal et al., 2025).

For mathematical data, the Curation Studio removes accidental repetition, which exceeds 1,000× in some widely used sources. It also scores examples for mathematical and reasoning value and synthesizes additional training material. These steps draw upon broader work on exact and semantic deduplication (Lee et al., 2022; Abbas et al., 2023; Tirumala et al., 2023), the diminishing value of repeated tokens (Muennighoff et al., 2023), model-based quality filtering (Li et al., 2024a; Penedo et al., 2024), and mathematical corpus construction and synthesis (Lewkowycz et al., 2022; Paster et al., 2024; Shao et al., 2024; Wang et al., 2024b; Han et al., 2024; Yang et al., 2024; Ben Allal et al., 2025; Zhou et al., 2025; Mahabadi et al., 2025).

For code data, a cascade of classifiers identifies programming language and data quality, followed by language-aware synthetic generation. Open code corpora use similar language identification and quality filtering (Kocetkov et al., 2023; Li et al., 2023a; Lozhkov et al., 2024; Guo et al., 2024; Hui et al., 2024; Huang et al., 2025b; Lozhkov et al., 2026), and several code models train on synthetic code (Rozière et al., 2023; Wei et al., 2024; Li et al., 2025).

Finally, Datology Curation Studio composes the processed sources into a general-purpose mixture using defaults derived from proxy-model experimentation. This follows the broader strategy of using smaller runs to estimate source interactions before committing to large-scale training (Xie et al., 2023a; Fan et al., 2023; Liu et al., 2025a; Ye et al., 2025; Chen et al., 2025b; Belenki et al., 2025; Diao et al., 2025).

Data provenance and decontamination. Both mixtures start from the same raw pool, and Datology Curation Studio generates the CURATED mixture's synthetic data from that pool, so neither mixture draws on a different raw corpus. Benchmark decontamination is also part of the treatment: Datology Curation Studio decontaminates the CURATED mixture against our evaluation data using 13-gram overlap, while the BASELINE uses the source datasets as released, with no decontamination beyond what their creators applied.

3 Pretraining Recipe

To measure the impact of data curation across scale, we designed a five-rung ladder of Qwen3-style mixture-of-experts (MoE) language models and a simple recipe for tuning and transferring training hyperparameters. The ladder scales from 117 million to 3.26 billion active parameters while preserving the principal architectural choices of Qwen3-30B-A3B (Yang et al., 2025a): 128 experts, top-8 routing, a head dimension of 128, grouped-query attention, and the Qwen3 residual and normalization conventions. The largest rung matches Qwen3-30B-A3B.

In pretraining, every hyperparameter tuning choice is both an opportunity to improve overall model quality and a risk of confounding our results by favoring one data setting over the other. Since our primary objective is to quantify the *relative* impact of data curation, the risks of confounding our results generally outweigh the benefits of better-tuned models, and we aim to minimize tuning more aggressively than is typical of modeling endeavors focused on delivering a single high-quality language model. Additionally, we tune our remaining pretraining hyperparameters on the BASELINE data and reuse them for CURATED training, so that any residual advantage from pretraining tuning favors the BASELINE.

Concretely, we simplify pretraining hyperparameter tuning in two ways. First, we remove hyperparameters: Hyperball constraints replace weight decay, and a single tied momentum coefficient replaces Adam's two β

values (§ 3.2). Second, scaling relationships motivated by linear-minimization-oracle (LMO) methods (§ 3.3) let us tune at small scales and short horizons and extrapolate to larger runs, which reduces the number and scale of tuning runs. We also adopt Warmup-Stable-Merge training (§ 3.4), which trades off a learning rate decay parameterization for a single weight averaging hyperparameter, f.

Notation. Let T denote the token budget, B the global batch in sequences per optimizer step, L_{seq} the sequence length, $B_{\text{tok}} = BL_{\text{seq}}$ the token batch, η the peak learning rate, and β a tied Adam coefficient ($\beta = \beta_1 = \beta_2$), with $\alpha = 1 - \beta$.

3.1 Model family

Table 2 summarizes our model scaling ladder. The active parameters column counts parameters used by one token, while the total parameters count includes all experts. All models train at sequence length 8,192; Appendix A.1 gives the hardware, software, per-rung parallelism, and typical throughput.

Table 2 Geometry of the Qwen3 MoE scaling ladder. All models use 128 experts with a top-8 routing strategy.

Rung	Layers	Hidden Dim	Expert FFN Dim	Q/KV heads	Active/total params
XS	9	384	128	6/2	117M/276M
S	15	640	256	10/2	254M/1.14B
M	24	1024	384	16/4	618M/4.02B
L	32	1536	576	24/4	1.43B/11.6B
XL	48	2048	768	32/4	3.26B/30.4B

3.2 Norm-constrained, equal-β optimization

AdamH replaces weight decay on representation matrices with explicit Hyperball constraints on the Frobenius norms of weights and updates (Wen et al., 2026). Weight decay acts indirectly on the equilibrium norm and therefore introduces another hyperparameter that can change the effective angular learning rate. Hyperball instead fixes each constrained matrix at radius

$$R = \|W_0\|_F, \tag{3.1}$$

reducing dimension-dependent variation in radial and angular update scales. This should improve transfer across widths and depths, although it does not make shape transfer exact.

We further tie the Adam coefficients, $\beta_1 = \beta_2$. Equal-beta Adam remains competitive with separately tuned coefficients across language-model settings (Orvieto and Gower, 2025), removes one search dimension, and eliminates a momentum-timescale mismatch term on the diagonal (Fernández-Hernández et al., 2026).

In our runs, Hyperball constraints apply to attention and expert matrices, with independent constraints on each expert's gate, up, and down projections. All other parameters use plain Adam, with a router learning-rate scale of 0.1 relative to the global learning rate, and no parameter group uses weight decay. We retain a standard Qwen3 initialization rather than the fan-in initialization used in the Hyperball optimization studies of Wen et al. (2026), so the radius in Eq. (3.1) remains initialization- and shape-dependent. We chose the router learning-rate scale and $\beta = 0.95$ from small-scale sweeps (Appendix A.3).

3.3 Learning-rate transfer across training horizons

Recent convergence proxies for linear-minimization-oracle (LMO) methods suggest joint scaling relationships among learning rate, batch size, and momentum for normalized SGD, signSGD, and Muon (Shulgin et al.,

2026). The underlying analysis holds model shape fixed, assumes a constant learning rate and simplified minibatch noise, and omits weight decay. We take the resulting laws for signSGD as priors for AdamH optimization of our MoE language models. Large-scale empirical studies of language modeling provide related evidence that optimal learning rate and batch size vary in this systematic way with dataset size (Filatov et al., 2025).

For token budget T, the LMO proxy gives

$$B(T) \propto \sqrt{T}, \qquad \eta(T) \propto \sqrt{\frac{B(T)}{T}}. \tag{3.2}$$

Starting from a tuned reference (η_0, B_0, T_0), moving along both relationships yields

$$B_1 = B_0\sqrt{\frac{T_1}{T_0}}, \qquad \eta_1 = \eta_0\left(\frac{T_0}{T_1}\right)^{1/4}. \tag{3.3}$$

For the horizon comparisons in this study, we use the fixed-batch, fixed-momentum transfer regime of Shulgin et al. (2026), which rescales only the learning rate:

$$B_1 = B_0, \qquad \alpha_1 = \alpha_0, \qquad \eta_1 = \eta_0\sqrt{\frac{T_0}{T_1}}. \tag{3.4}$$

While the batch size may be larger than optimal at the shorter horizons, this simplifies our tuning work in practice by allowing us to change only the learning rate and keep the batch size fixed to a setting for which we have already identified an efficient distributed training configuration.

To apply these relationships, we normalize training horizons by tokens per active parameter:

$$T = cN_{\text{active}}, \qquad K = \left\lceil \frac{T}{B_{\text{tok}}} \right\rceil, \tag{3.5}$$

where c is the number of tokens per active parameter and K is the number of optimizer steps.

We use three horizons: a full-c horizon sets the XL rung's token budget to one trillion tokens, corresponding to $c_{\text{full}} \simeq 307.12$, and two shorter-horizon settings of $c = 80$ and $c = 160$ reserved for development and hyperparameter tuning. Table 3 lists the resulting token budgets for every rung and horizon.

Table 3 Derived token budgets for the three experimental horizons.

Rung	Active parameters	$c = 80$ tokens	$c = 160$ tokens	Full-c tokens
XS	117M	9.33B	18.7B	35.8B
S	254M	20.3B	40.6B	78.0B
M	618M	49.5B	98.9B	190B
L	1.43B	115B	229B	440B
XL	3.26B	260B	521B	1T

Tuning order. In this analysis, the optimal learning rate depends on both batch size and momentum, so we fix those two first and tune the learning rate last. Fixing momentum follows the regime Shulgin et al. (2026) identify as most practically relevant, and it removes a tuning dimension.

We first fix the batch size at each rung for hardware throughput, adjusting the batch sizes given by Eq. (3.3) to parallelization-friendly values and using Eq. (3.2) to correct the learning rate for these adjustments. We then select β from joint learning-rate and β sweeps at small scale (§ 3.2 and Appendix A.3). With batch size and β fixed, we tune the learning rate with short-horizon $c = 80$ sweeps at the XS and S scales and transfer it across horizons with the fixed-batch rule in Eq. (3.4), which sets the $c = 160$ and $c = 80$ rates to 1.385× and

1.960× the full-horizon rate (Eq. (A.2)). We validated these rates by scaling them up and down by $\sqrt{2}$ at various horizons across XS, S, and M. After these checks succeeded, we transferred the learning rates to L and XL without further tuning, avoiding expensive tuning runs at the largest scales. Appendix A.3 lists the resulting AdamH learning-rate selections for every rung and horizon.

3.4 Warmup-Stable-Merge training

We adopt a Warmup-Stable-Merge training schedule for two desirable properties: (1) it avoids challenges in further rounds of training associated with learning rate decay, and (2) it provides more flexibility towards a horizon-free paradigm.

Warmup-Stable-Merge replaces horizon-dependent learning-rate decay with constant-rate training followed by principled model averaging (Tian et al., 2025). Related anytime-training results show that horizon-free learning rates paired with weight averaging can track the performance envelope of schedules tuned to a predetermined endpoint (Meterez et al., 2026). We use the online averaging rule of Meterez et al. (2026):

$$A_1 = W_1, \qquad A_t = \rho_t A_{t-1} + (1-\rho_t) W_t \ (t>1), \qquad \rho_t = 2^{-f/t}, \tag{3.6}$$

where t is the time step, W_t is the current parameter tensor, and hyperparameter f controls the averaging horizon. Under this parameterization, the averaging becomes more diffuse as training progresses, and larger values of f increase the temporal concentration of the averaging at all steps.

All runs use 1,000 warmup steps followed by a constant peak learning rate. For XS through L, we perform online weight merging with Eq. (3.6) and $f = 50$. At XL scale, however, $f = 50$ caused a collapse in model quality. We hypothesize that the parameterization of Meterez et al. (2026) may need to be normalized by factors such as batch size to more effectively transfer across experiment scales. For the XL runs, we instead set $f = 250$ and merged checkpoints taken every 2,000 steps offline, replacing the $f = 50$ online-merged checkpoints.

3.5 Routing stabilization through auxiliary loss

The routing configuration follows Qwen3's global-batch load-balancing principle (Yang et al., 2025a). Learned top-8 routing is trained with

$$\mathcal{L}_{\text{LBL}}^{(\ell)} = \frac{\lambda_{\text{job}}}{n_{\text{MoE}}} E \sum_{e=1}^{E} f_e p_e, \tag{3.7}$$

where $E = 128$, f_e is expert e's assignment frequency, and p_e is its mean router probability. We use $\lambda_{\text{job}} = 5 \times 10^{-2}$, divided evenly over the n_{MoE} layers. Parameterizing a fixed weight for auxiliary loss across all layers, rather than using a per-layer term, allowed a single hyperparameter setting to work smoothly across all model sizes. Assignment statistics are aggregated over data-parallel ranks and all gradient-accumulation microbatches in an optimizer step.

3.6 Tokenizer

All models in the ladder use the DeepSeek-V4 tokenizer, a byte-level BPE vocabulary of 129,280 tokens. We chose it after comparing five widely used tokenizers: DeepSeek-V4, Qwen3, Mistral's Tekken, Llama 3, and OpenAI's o200k. In matched small-scale training runs that differed only in the tokenizer, DeepSeek-V4 consistently performed best, with the clearest advantage on code. We suspect this is because it splits identifiers, operators, and indentation more cleanly along the structure of source code.

Every token budget in this paper is counted in DeepSeek-V4 tokens, so the horizons in § 3.3 are specific to this tokenizer. The vocabulary size also determines the size of the embedding and output layers, which are included in the total parameter counts in Table 2.

4 Context Extension

To test whether the pretraining-data advantage persists beyond pretraining, we carry every BASELINE–CURATED model pair through matched context extension and post-training. We extend every model from its 8k pretraining context to 64k in a single stage, following the same procedure at all scales. Each model undergoes continued pretraining for 10 billion tokens at 64k sequence length on an 85/15 token blend of replayed pretraining data and intentionally selected long documents. We extend the context with YaRN (Peng et al., 2024), which rescales the rotary frequencies by a factor of 8 (8k to 64k context lengths), and keep RoPE θ fixed at its pretraining value of 500,000. We set our data loader to use best-fit packing in the style of Ding et al. (2024) to ensure that long documents are minimally truncated over multiple contexts at training time. Following Zhao et al. (2024) and Team Olmo et al. (2025), we disable cross-document attention during the context extension phase. We tune the learning rate per model under a Warmup-Stable-Decay schedule (Hu et al., 2024) with 40 warmup steps, a 20% decay phase, and a minimum learning rate factor of 0.05. We then merge the resulting 64k model with the starting 8k checkpoint to better preserve short-context capabilities.

Across both the M-scale and the L-scale models, we found that the best extension learning rate fell below the pretraining peak learning rate, allowing us to narrow the search space at XL scale. We tuned the mixture at M scale, where 85/15 and 67/33 token mixes performed comparably on RULER, and chose the 85/15 mix for all runs to reduce the risk of forgetting.

Long-context data. To ensure a fair comparison between the CURATED and BASELINE data, the set of long documents remains fixed for all models (while the replay data comes from the respective pretraining distribution). Specifically, we use the Longmino dataset from OLMo 3, described in Team Olmo et al. (2025). The Longmino data contains both high-quality natural PDF-derived text and synthetic augmentations of science PDF data. We mixed the sources available in the Dolma3 Longmino Pool to create a mixture similar to the Dolma3 Longmino 50B Mix, replacing the OLMo 3 midtraining data with our own replay data.

Tracking performance during training. We evaluate on RULER (Hsieh et al., 2024) at 4k, 8k, 16k, 32k, and 64k and on a general evaluation subset as a proxy for how much the model forgets during context extension. The general evaluation subset contains nine short-context tasks covering code, math, exam-style reasoning, and knowledge (Appendix C.2).

In our experiments, we find that long-context retrieval improves rapidly during the first 2.5 billion tokens of extension, after which the rate of improvement slows. Short-context capability initially regresses during context extension and then recovers to slightly below its pretraining level. After evaluating several learning rates, we find that the first 2.5 billion tokens provide sufficient signal to select the extension learning rate using short sweeps.

Model merging benefit. We merge checkpoints by taking a weighted average of the parameter values using `mergekit` (Goddard et al., 2024):

$$\theta_{\text{merged}} = \theta_{8k} + w\,(\theta_{64k} - \theta_{8k}), \tag{4.1}$$

where w is the weight on the context-extended model that we sweep per model.

Merging also improves math performance after SFT. Applying the same SFT recipe to the L BASELINE initialized from the merged long-context checkpoint improves average math accuracy by 2.6 points compared with initializing from the unmerged checkpoint, with most of the gain on MATH-500, AIME 2024, and GSM8K (Table 4).

5 Post-Training

To test whether the gains from pretraining data curation persist through post-training, we apply a three-stage recipe that produces thinking-style models. First, we apply SFT. We then train four RLVR specialists, one

Table 4 Post-SFT math scores for the L BASELINE, starting SFT from the un-merged or the merged 64k checkpoint. Both use the same SFT recipe (9,536 steps, ≈20B tokens). Metrics match Table 12; sampled tasks report pass@1.

Task	Un-merged	Merged	Δ
GSM8K	48.60	50.87	+2.27
MATH-500	55.00	60.60	+5.60
AIME 2024	5.52	9.06	+3.54
AIME 2025	11.67	12.50	+0.83
AMC23	29.45	30.31	+0.86
Mean	30.05	32.67	+2.62

each for mathematics, code, instruction following, and tool use, before merging them into a single model. We apply this recipe to the M, L, and XL models; the smaller rungs fall below the noise threshold of several post-training evaluations. Unlike pretraining, whose token budget scales with model size (§ 3.3), post-training uses the same fixed budget at every scale.

Unless otherwise stated, the BASELINE and CURATED models receive the same post-training data and matched training recipes. When model-specific selection is required, both models in a pair receive the same candidate settings and selection criterion, so both get the same tuning opportunity.

5.1 SFT

Starting from each merged 64k checkpoint (§ 4), we train with SFT for 9,536 steps on 64k-token sequences built by greedy packing. We use a global batch of 32 sequences, corresponding to approximately 2.1 million tokens per step and 20 billion tokens in total, with loss applied to every assistant turn. Conversations longer than the context window are dropped rather than truncated; an earlier run trained on truncated traces produced non-terminating reasoning at evaluation time. Appendix B.1 gives the optimizer settings.

SFT data. We construct the SFT mixture by building on the data used to train OLMo 3 (Team Olmo et al., 2025), specifically its thinking-model variant, Dolci-Think. Ablations identified weaknesses in code and tool-calling evaluations, so we add coding-assistant data from sources including UltraData SFT, OpenCoder (Huang et al., 2025b), and Glaive Code Assistant, together with tool-calling data from ToolMind (Yang et al., 2025b). We validate the mixture through data ablations across several open-source base models and the BASELINE M model. Although we apply no additional curation at this stage, we perform light preprocessing: we remove UltraData code rows with malformed reasoning tags or exact duplicates and normalize nonstandard tool schemas in ToolMind.

5.2 RLVR

After SFT, we apply RLVR (Lambert et al., 2025) in four domains: mathematics, code, instruction following, and tool use. From each BASELINE and CURATED SFT checkpoint at the M, L, and XL scales, we train four independent specialists, one per domain. All specialists share a common Group Relative Policy Optimization (GRPO) recipe (Shao et al., 2024) and differ only in their training data, reward function, and a small number of scale-dependent settings. Below, we describe the shared recipe, the domain-specific settings, and how we merge the four specialists into one model per scale and pretraining data setting. Section 6 evaluates the merged models, and § 7.4 reports the gains of the individual specialists.

Shared recipe. Each specialist trains for 100 steps with GRPO in an asynchronous PRIME-RL setup (Intellect, 2025). Answers cut off by the token limit receive zero reward, which becomes important for the XL mathematics run below. Two domains needed reward design beyond a programmatic check. Instruction-

Table 5 SFT data mix. Shares are fractions of training tokens.

Source	Share	Contribution
Reasoning		
Dolci-Think	84%	Long chain-of-thought across math, code and general reasoning
Code		
UltraData code (think)	5%	Code with reasoning traces
UltraData code (no-think)	2%	Direct code answers
Glaive code assistant	3.25%	Coding-assistant conversations
Evol-Instruct code	0.45%	Code instructions of increasing complexity
McEval	0.3%	Code instructions in languages beyond Python
Tool use		
ToolMind	5%	Multi-turn tool calls, including parallel calls

Table 6 RLVR training data, reward, and settings for each domain (IF: instruction following; LR: learning rate). Group is the number of answers per prompt; share is the fraction of a domain's training prompts.

Domain	Group	Max tokens	LR M, L	LR XL	Source	Share	Reward
Math	8	28,672	4e-6	2e-6	BigMath	100%	Boxed answer matches reference
Code	8	16,384	2e-6	2e-6	OpenCodeInstruct	72%	Fraction of tests passed
					I3	18%	Test-based checker
					CodeIO	10%	Predicted output matches
IF	8	12,288	4e-6	2e-6	IFEvalG (easier)	100%	Constraint fraction, judge-gated
Tool use	16	28,672	4e-6	4e-6	ToolACE	38%	Matched-call fraction
					xLAM	34%	Matched-call fraction
					Hammer (masked)	13%	Matched-call fraction
					No-tool	15%	Abstains without a call

following constraint checks alone are easy to game, and this was the domain most prone to earning reward without a useful answer, so a large language model (LLM) judge zeroes the reward for empty, repetitive, or claim-only responses. Tool-use training includes no-tool examples that reward declining to call any tool. Appendix B.2 gives the full shared recipe and the reward definitions.

Domains. Table 6 lists the training data, reward, and main settings for each domain. All training sets are pinned DatologyAI revisions of BigMath (Albalak et al., 2025), OpenCodeInstruct (Ahmad et al., 2025), I3 (Team et al., 2025), CodeIO (Li et al., 2025), IFEvalG (Zhou et al., 2023), ToolACE (Liu et al., 2025b), xLAM (Liu et al., 2024), and Hammer (Lin et al., 2024). Rewards are computed by programmatic checks, with an LLM judge additionally gating instruction-following rewards.

Scale-dependent settings. M and L use the same learning rates, and at XL scale we halve the rate for mathematics and instruction following (Table 6). One setting differs between the BASELINE and CURATED models. Under the shared recipe, the CURATED XL mathematics run collapsed within 40 steps: truncation rose from 16% to 88%, and held-out accuracy fell from 36% to 8%. A lower learning rate only delayed the collapse. Adding DAPO's soft overlong punishment (Yu et al., 2025) with a maximum penalty of 0.25 restored stable training, but the same penalty reduced the gains of the BASELINE XL mathematics run. We therefore sweep the penalty weight over $\{0, 0.25\}$ for both runs and retain the better setting for each: 0.25 for CURATED and 0 for BASELINE.

Model merging. We combine the four specialists into one model per scale and pretraining data setting using task arithmetic (Ilharco et al., 2023). Starting from the shared SFT parent θ_0, the merged model is $\theta = \theta_0 + \frac{\lambda}{4}\sum_{i=1}^{4}(\theta_i - \theta_0)$, that is, λ times the mean of the four task vectors, where the θ_i are the specialist weights and $\lambda = 0.7$. We compared several merging recipes on the BASELINE models, and this one transferred best across model scales, so we applied it unchanged to all six models. Merging costs some instruction-following performance: every specialist except the instruction-following specialist regresses on instruction-following benchmarks, and equal weighting dilutes that specialist's contribution. Section 7.4 compares the merged BASELINE and CURATED models.

6 Evaluation

We evaluate the effect of pretraining data curation at four points in the training pipeline: after pretraining, after context extension, after supervised fine-tuning (SFT), and after merging the RLVR specialists. We refer to models by their rung names, XS through XL (Table 2).

We evaluate predictive quality and task performance across the main capabilities targeted by the pretraining mixture. BPB (§ 2.2) measures how well models predict reference answers, while benchmark scores measure their ability to select or generate correct responses.

For base models, the benchmark suite spans knowledge and reasoning (MMLU, MMLU-Pro, AGIEval, BBH, and a broad set of question-answering, reading-comprehension, and commonsense benchmarks), mathematics (from grade-school word problems to competition problems and robustness to problem perturbations), code (function generation, multilingual and library-heavy generation, execution reasoning, code understanding, and repository-level completion), and multilingual understanding. After context extension, we add long-context retrieval with RULER (Hsieh et al., 2024) and repeat the base-model evaluations to assess retention of short-context capabilities. For SFT and RLVR models, we keep the same capability areas and add harder reasoning and competition benchmarks (GPQA Diamond, AIME 2024 and 2025, HMMT 2025, LiveCodeBench), instruction following and writing quality (IFEval, IFBench, Arena-Hard Writing), and tool use (BFCL, Nexus, SEAL). Appendix C.1 lists every benchmark with its source, and the results tables report the metric used at each training stage.

Before any training runs, we audited the code benchmarks we report and corrected flawed prompts, reference solutions, and tests, producing HumanEval++, MBPP++, and MultiPL-E+; Appendix C.3 describes the fixes.

Base models. We report BPB on the four-category suite defined in § 2.2. Task evaluations use task-specific prompts, demonstrations, and scoring procedures, with the few-shot settings specified by each benchmark configuration; Appendix C.2 gives representative settings. We retain the metric associated with each evaluation, including multiple-choice accuracy, exact match, F1, and code-execution success. For sampled tasks, the main aggregates use pass@1.

Context-extended models. We evaluate context-extended models for long-context retrieval and retention of short-context capabilities. RULER is evaluated at 4k, 8k, 16k, 32k, and 64k tokens, with results reported by length and averaged equally across the five lengths. During training, we monitor retention on a smaller nine-task subset (Appendix C.2). After training, we evaluate the final checkpoints on the full pretraining benchmark suite. We also report BPB on the same suite as for the pretrained models.

Post-trained models. We evaluate both the SFT checkpoints and the final models obtained by merging RLVR specialists. Within each benchmark, the BASELINE and CURATED models are evaluated with matching prompts, decoding settings, and scoring procedures (Appendix C.2). We report pass@1 for code generation

and average CRUXEval's input- and output-prediction scores.[3] Multilingual macro-averages give equal weight to the included languages. IFEval and IFBench use prompt-level loose accuracy. Tool-use results retain the benchmark-specific accuracy and F1 metrics shown in the tables. The final post-training aggregate is the mean of 25 benchmark scores on a 0–100 scale, using pass@1 where multiple pass rates are available. It includes every row of Table 14 except SEAL's format and tool checks, which are fail-fast criteria that gate SEAL's parameter-F1 score rather than separate benchmarks.

7 Results

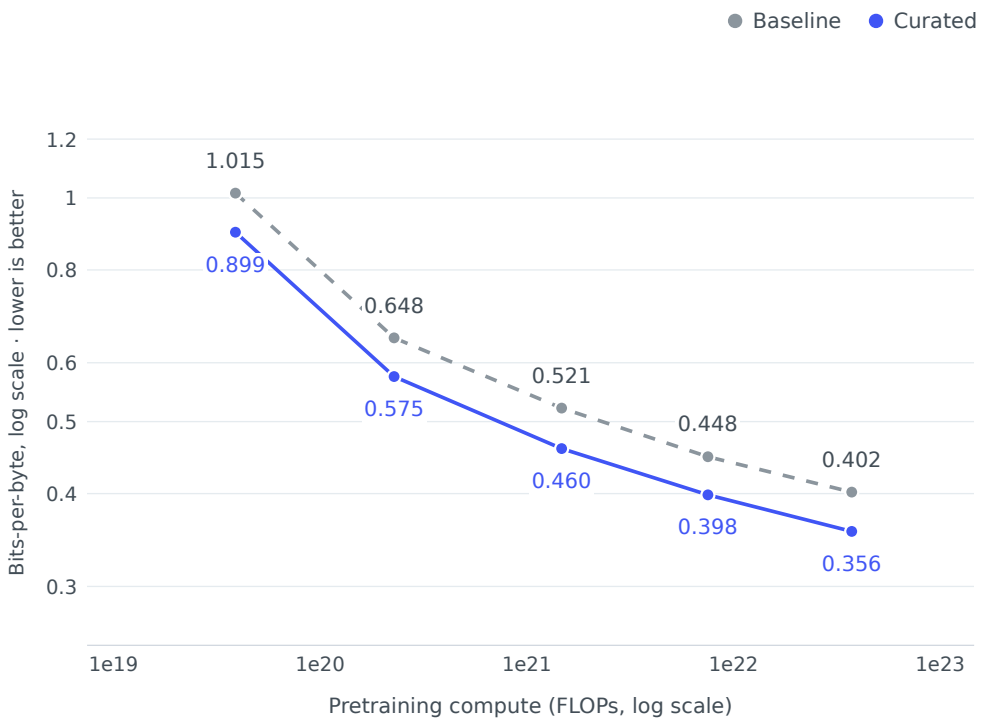


Figure 3 Overall BPB (lower is better; log scale) for the XS–XL models. Curation lowers BPB by 11–12% at every scale, across pretraining budgets from 3.9×10^{19} to 3.7×10^{22} FLOPs.

We organize the results around the three research questions posed in the introduction:

- Scale: How does the benefit of data curation change with model size and pretraining compute?
- Persistence: Does the gap shrink, hold, or grow through training stages?
- Capabilities: Where are the gains largest, where are they modest, and where does performance regress?

7.1 Curation lowers BPB at every scale

Curation reduces overall BPB at all five model scales, with relative improvements of 11.1–11.8% (Figure 3). This advantage holds across nearly three orders of magnitude in pretraining compute, from 3.9×10^{19} to 3.7×10^{22} FLOPs (Appendix A.2). At M scale, overall BPB falls from 0.521 to 0.460; at XL scale, it falls

[3] CRUXEval scores are sensitive to prompt format, particularly for weaker checkpoints. We evaluate every checkpoint with the same configuration, so scores are comparable within this report but not directly comparable to CRUXEval results reported elsewhere.

Model	Data	Code	Math	Analytical	Knowledge	Overall
XS	Baseline	1.082	1.038	0.970	0.969	1.015
	Curated	**0.880**	**0.855**	**0.900**	**0.963**	**0.899**
S	Baseline	0.690	0.635	0.613	0.655	0.648
	Curated	**0.613**	**0.515**	**0.542**	**0.629**	**0.575**
M	Baseline	0.563	0.499	0.489	0.534	0.521
	Curated	**0.510**	**0.400**	**0.425**	**0.504**	**0.460**
L	Baseline	0.492	0.425	0.419	0.458	0.448
	Curated	**0.452**	**0.352**	**0.358**	**0.431**	**0.398**
XL	Baseline	0.455	0.378	0.368	0.407	0.402
	Curated	**0.407**	**0.327**	**0.313**	**0.375**	**0.356**

Table 7 Pretraining BPB scores by domain across five model scales. Lower is better; bold indicates the lower score within each model pair.

from 0.402 to 0.356. The absolute gap becomes smaller between these two scales as loss falls for both models, but the proportional benefit is essentially unchanged. Curation is not a small-scale effect.

The domain breakdown shows lower BPB in every domain at every scale (Table 7). At M scale, relative reductions are 19.8% in mathematics, 13.2% in analytical reasoning, 9.4% in code, and 5.6% in knowledge. At XL scale, mathematics and analytical reasoning improve by 13.5% and 14.8%, respectively, compared with 10.4% in code and 7.7% in knowledge. At both scales, mathematics and analytical reasoning gain the most, and knowledge gains the least.

7.2 Gains carry over to task performance

The largest task gains are in mathematics, program generation, and several knowledge and reasoning benchmarks (Table 8). At XL scale, curation improves MATH-500 by 16.80 points, HumanEval++ pass@1 by 20.76 points, and ARC-Challenge by 22.10 points.

Several gains are large enough for the curated L model to exceed the XL baseline. It scores 54.60 versus 47.60 on MATH-500, 32.77 versus 29.79 on HumanEval++, and 65.78 versus 53.75 on ARC-Challenge, despite using approximately one-fifth of the pretraining compute. This is not a general equivalence between the two models: the smaller curated model trails the larger baseline on 16 of the 49 pretraining benchmarks, including AGIEval, BBH, DROP, PIQA, COPA, MBPP++, CRUXEval, and factual question answering.

The clearest recurring weakness is in factual recall. Curated models score below their baseline counterparts on TriviaQA short answer and Natural Questions at all three scales.

7.3 The gap survives context extension

Context extension provides the first test of persistence. On the same BPB suite, the relative advantage remains 11–12%, close to its pretraining value. Overall BPB is 0.460 versus 0.516 at M scale, 0.397 versus 0.449 at L scale, and 0.356 versus 0.401 at XL scale for curated and baseline models, respectively (Table 9). The curated models retain lower BPB in all four domains, so context extension leaves the predictive-quality gap intact.

At the task level, the XL curated model retains large advantages on MATH-500 (65.60 versus 48.40), HumanEval++ (47.50 versus 29.85), and BigCodeBench (32.35 versus 19.75) (Table 11). The weaker factual-

		M		L		XL	
Benchmark	**Metric**	**Baseline**	**Curated**	**Baseline**	**Curated**	**Baseline**	**Curated**
Knowledge and reasoning							
MMLU	Acc.	32.96	**40.99**	32.14	**50.74**	49.05	**57.98**
MMLU-Pro	Macro	12.50	**14.70**	11.79	**24.36**	20.92	**32.18**
AGIEval English	Acc.	26.58	**31.20**	29.08	**38.76**	38.92	**44.55**
BBH CoT	EM	29.75	**32.39**	30.27	**40.67**	41.51	**54.26**
ARC-Challenge	Acc.	30.03	**46.50**	30.55	**65.78**	53.75	**75.85**
CommonsenseQA	Acc.	22.19	**27.19**	25.31	**52.83**	51.43	**62.65**
DROP	F1	**28.30**	26.67	32.14	**33.15**	42.44	**45.51**
ARC-Easy	Acc.	29.76	**61.07**	34.47	**83.88**	74.75	**90.49**
Natural Questions	F1	**11.34**	9.95	**13.77**	11.91	**18.08**	15.18
Social IQA	Acc.	36.95	**42.78**	40.12	**51.94**	53.63	**59.62**
SciQ	Acc/char	37.30	**69.00**	41.90	**85.70**	81.20	**89.90**
PIQA	Acc.	49.62	**54.95**	49.78	**64.53**	**69.04**	68.39
RACE	Acc.	**32.92**	32.73	35.12	**35.50**	**34.64**	34.35
OpenBookQA	Acc.	33.00	**47.80**	33.00	**62.40**	61.40	**73.20**
Winogrande	Acc.	**50.36**	48.30	50.04	**51.62**	50.51	**53.04**
BoolQ	Acc.	**61.65**	37.77	40.52	**63.33**	53.55	**77.13**
COPA	Acc.	71.00	**73.00**	**78.00**	74.00	**80.00**	75.00
TriviaQA short answer	EM	**20.47**	17.84	**32.89**	27.76	**49.18**	40.85
Mathematics							
GSM8K	EM	18.95	**30.63**	37.07	**52.39**	45.34	**59.97**
GSM-Symbolic	EM	15.78	**25.46**	33.34	**47.34**	37.96	**54.82**
MATH-500	EM-flex	19.20	**28.00**	34.20	**54.60**	47.60	**64.40**
DAT Word Math	Math verify	7.50	**16.41**	20.16	**43.28**	32.03	**52.19**
AMC23	EM-flex	2.50	**12.50**	2.50	**27.50**	27.50	27.50
College Math	Math verify	15.37	**29.03**	28.74	**42.55**	39.18	**47.94**
MathPerturb – hard	EM (perturb.)	2.87	**4.30**	9.68	**18.64**	15.77	**29.75**
MathPerturb – simple	EM (perturb.)	6.45	**9.68**	17.92	**29.03**	34.05	**50.90**
Code							
HumanEval++	pass@1	16.22	**23.96**	26.04	**32.77**	29.79	**50.55**
MBPP++	pass@1	13.33	**28.62**	38.03	**45.93**	51.03	**61.49**
MultiPL-E+	Macro pass@1	4.91	**14.11**	7.15	**20.97**	11.36	**30.01**
BigCodeBench	pass@1	4.25	**10.32**	10.91	**19.95**	18.18	**30.28**
DS-1000	pass@1	4.40	**5.30**	4.92	**11.52**	9.20	**13.80**
CRUXEval	pass@1	8.61	**10.27**	**15.03**	14.16	16.46	**17.39**
CodeMMLU	EM	**18.48**	12.11	12.78	**34.74**	17.53	**30.87**
RepoBench Python cross-file first	EM	7.98	**9.88**	10.11	**12.35**	13.10	**14.88**
RepoBench Python cross-file random	EM	24.01	**26.06**	27.28	**29.33**	28.37	**31.19**
RepoBench Python in-file	EM	22.09	**23.68**	25.65	**26.61**	27.24	**29.09**
RepoBench Java cross-file first	EM	8.26	**12.68**	10.17	**16.64**	12.81	**19.68**
RepoBench Java cross-file random	EM	19.54	**24.95**	23.19	**30.01**	26.14	**33.45**
RepoBench Java in-file	EM	29.02	**34.60**	34.01	**37.82**	36.05	**41.76**
RepoEval line	EM	16.81	**17.13**	18.63	**20.44**	20.19	**22.06**
RepoEval API	EM	13.56	**15.31**	17.13	**17.50**	18.25	**19.94**
RepoEval function	Edit sim.	**29.95**	29.94	31.06	**32.91**	32.71	**34.69**
Multilingual							
MMLU	Acc.	28.89	**33.73**	28.61	**36.59**	32.34	**40.26**
MMMLU	EM	**17.03**	15.47	15.17	**15.66**	**16.28**	15.36
ARC-Challenge Multilingual	Acc.	28.90	**33.59**	29.57	**44.29**	34.32	**52.08**
Belebele (MC)	Acc.	34.41	**37.37**	37.75	**45.33**	41.51	**53.79**
MMLU	Acc/char	26.95	**28.09**	27.55	**29.51**	28.73	**31.71**
ARC-Challenge Multilingual	Acc/char	25.51	**27.64**	27.20	**31.56**	29.25	**36.74**
Belebele (RC)	Acc/char	33.18	**33.71**	36.92	**38.83**	39.98	**42.17**

Table 8 Pretrained model results grouped by capability. Scores are on a 0–100 scale. Bold indicates the higher score within each BASELINE–CURATED pair.

Table 9 Category-average BPB on the evaluations from § 2.2 for the context-extended models (lower is better), using the same checkpoints as Table 10.

	M		L		XL	
Category	Baseline	Curated	Baseline	Curated	Baseline	Curated
Overall	0.516	0.460	0.449	0.397	0.401	0.356
Math	0.494	0.401	0.426	0.351	0.379	0.325
Code	0.554	0.505	0.485	0.444	0.449	0.407
Analytical	0.488	0.429	0.424	0.361	0.369	0.314
Knowledge	0.529	0.506	0.461	0.432	0.408	0.376

recall results also persist: TriviaQA is 40.61 versus 51.08 and Natural Questions is 14.28 versus 17.57. Both the positive and the negative gaps carry through context extension.

Long-context retrieval, by contrast, shows no consistent difference (Table 10). Mean RULER scores differ by less than two points at every scale, and the CURATED model is ahead at the M and L scales but behind at XL scale. This is not surprising given that we apply the same context extension recipe to both models in a pair (§ 4): CURATED pretraining neither helps nor hurts the long-context retrieval that the recipe produces. Context extension therefore preserves the CURATED advantage without adding one of its own; a long-context gap appears only after post-training (§ 7.4).

Table 10 RULER for the context-extended models (higher is better). ≤8k is the mean of the 4k and 8k lengths; 16k–64k is the mean of the 16k, 32k and 64k lengths.

	M		L		XL	
RULER	BASELINE	CURATED	BASELINE	CURATED	BASELINE	CURATED
Mean	55.50	56.44	57.62	59.48	69.19	67.50
≤8k	64.83	68.38	68.47	70.87	80.00	76.55
16k–64k	49.28	48.48	50.38	51.89	61.98	61.47

7.4 The gap survives post-training

Supervised fine-tuning. SFT preserves the broad advantage in mathematics and code (Table 12). At XL scale, the advantages include 12.40 points on MATH-500, 26.67 on AIME 2024, and 18.72 on HumanEval++. Because base-model and SFT evaluations use different prompts and decoding, these gaps are not directly comparable to the pretraining gaps.

CURATED models also lead on general knowledge and reasoning, outperforming their BASELINE counterparts on AGIEval English and MMLU-Pro at all three scales. At XL scale, AGIEval English increases from 41.77 to 50.15, while MMLU-Pro increases from 23.47 to 33.27.

These results also address a caveat on the base-model multiple-choice gains: base-model accuracy depends partly on whether a model has learned the answer format (§ 2.2), so part of those gains could reflect format familiarity rather than knowledge. SFT trains both models on the same data and formats, and the gap remains. On MMLU-Pro, the CURATED models lead by 5.6, 11.6, and 9.8 points after SFT at the M, L, and XL scales, compared with 2.2, 12.6, and 11.3 points before post-training (Table 8).

The advantage does not transfer uniformly to instruction following. CURATED models improve on Arena-Hard Writing at all three scales. IFEval improves at the M and L scales, including from 53.60 to 57.49 at L scale, with a small decline at XL scale from 64.51 to 62.48.

Tool-use gains are consistent across the three scales on BFCL live, BFCL non-live, and Nexus. At XL scale, BFCL live accuracy improves from 51.53 to 59.00, non-live accuracy from 67.48 to 75.11, and Nexus from 38.99 to 42.45. SEAL parameter F1 improves at every scale, including from 78.50 to 82.22 at XL scale, while format accuracy is above 99% for all models, so the gains come from argument correctness rather than formatting.

On RULER 32k, which post-trained models run in a thinking format, the SFT checkpoints show a long-context advantage that was absent after context extension (Tables 12 and 14). CURATED models score 19.03, 28.72, and 32.30 at M, L, and XL scale, compared with 13.81, 20.27, and 23.13 for the corresponding BASELINE models. In the outputs, the BASELINE models lose more long-context retrieval during post-training: they more often end an answer prematurely, exhaust the thinking budget, or recover fewer of the hidden items. Because BASELINE and CURATED models share matched post-training recipes and identical evaluation settings, we attribute the gap to differences in how the context-extended checkpoints hold up under post-training rather than to a difference in long-context capability after extension.

		M		L		XL	
Benchmark	**Metric**	BASELINE	CURATED	BASELINE	CURATED	BASELINE	CURATED
Knowledge and reasoning							
MMLU	Acc.	33.83	**40.55**	34.01	**50.31**	49.88	**58.06**
MMLU-Pro	Macro	12.53	**13.92**	11.91	**24.89**	22.21	**31.00**
AGIEval English	Acc.	24.86	**31.32**	30.10	**38.44**	39.66	**44.57**
BBH CoT	EM	28.78	**31.64**	32.17	**41.39**	41.77	**53.30**
ARC-Challenge	Acc.	29.61	**47.35**	34.73	**64.76**	56.40	**76.71**
CommonsenseQA	Acc.	22.93	**28.99**	27.03	**52.01**	51.27	**61.92**
DROP	F1	**27.29**	26.30	31.00	**32.72**	43.66	**45.70**
ARC-Easy	Acc.	34.85	**61.03**	46.89	**82.41**	75.63	**90.40**
Natural Questions	F1	**9.82**	9.23	**13.24**	11.02	**17.57**	14.28
Social IQA	Acc.	34.80	**42.63**	40.63	**52.66**	52.92	**59.72**
SciQ	Acc/char	38.80	**68.90**	50.70	**85.90**	79.20	**89.30**
PIQA	Acc.	48.86	**54.90**	48.69	**64.53**	68.39	**69.31**
RACE	Acc.	32.63	**33.11**	**35.31**	35.12	34.74	**35.22**
OpenBookQA	Acc.	33.00	**46.20**	37.80	**59.80**	60.00	**70.80**
Winogrande	Acc.	**50.51**	48.46	49.64	**51.38**	51.54	**54.30**
BoolQ	Acc.	**57.46**	37.49	43.15	**63.98**	41.68	**76.61**
COPA	Acc.	**74.00**	72.00	**77.00**	73.00	**81.00**	72.00
TriviaQA short answer	EM	**21.11**	17.25	**33.10**	27.77	**51.08**	40.61
Mathematics							
GSM8K	EM	17.66	**27.14**	34.27	**50.11**	45.72	**56.41**
GSM-Symbolic	EM	14.06	**22.18**	29.04	**44.18**	37.92	**52.54**
MATH-500	EM-flex	20.20	**22.60**	31.00	**50.00**	48.40	**65.60**
DAT Word Math	Math verify	9.84	**20.62**	19.84	**42.81**	33.28	**59.06**
AMC23	EM-flex	5.00	**12.50**	15.00	**17.50**	**35.00**	32.50
College Math	Math verify	17.99	**27.93**	28.25	**40.81**	39.78	**48.97**
MathPerturb – hard	EM (perturb.)	**3.23**	2.87	6.09	**15.41**	14.34	**26.88**
MathPerturb – simple	EM (perturb.)	**11.47**	9.68	20.43	**31.90**	33.33	**49.10**
Code							
HumanEval++	pass@1	16.77	**23.72**	24.51	**33.66**	29.85	**47.50**
MBPP++	pass@1	**24.27**	8.94	36.60	**49.71**	50.34	**61.72**
MultiPL-E+	Macro pass@1	4.26	**13.01**	6.03	**19.65**	11.00	**28.29**
BigCodeBench	pass@1	5.49	**8.49**	11.53	**20.74**	19.75	**32.35**
DS-1000	pass@1	4.50	**5.32**	6.00	**12.36**	10.86	**14.74**
CRUXEval	pass@1	8.28	**9.76**	**16.07**	14.12	**19.13**	16.12
CodeMMLU	EM	**19.41**	10.93	14.87	**36.20**	21.68	**36.27**
RepoBench Python cross-file first	EM	8.67	**10.74**	11.37	**13.38**	12.98	**14.93**
RepoBench Python cross-file random	EM	24.26	**26.64**	27.60	**29.65**	29.27	**32.67**
RepoBench Python in-file	EM	22.60	**25.02**	26.10	**27.37**	27.50	**30.36**
RepoBench Java cross-file first	EM	8.52	**13.28**	12.09	**18.23**	14.40	**20.21**
RepoBench Java cross-file random	EM	19.47	**26.14**	23.05	**29.16**	28.25	**34.22**
RepoBench Java in-file	EM	29.88	**35.65**	34.93	**37.56**	36.64	**42.88**
RepoEval line	EM	15.94	**17.69**	18.56	**19.56**	20.88	**22.62**
RepoEval API	EM	13.31	**14.75**	**17.12**	15.88	18.19	**19.19**
RepoEval function	Edit sim.	30.47	**30.65**	32.18	**32.56**	32.78	**33.95**
Multilingual							
MMLU	Acc.	28.12	**32.79**	29.25	**36.69**	33.89	**41.06**
MMMLU	EM	**18.01**	15.79	14.86	**16.11**	**17.24**	16.03
ARC-Challenge Multilingual	Acc.	28.17	**32.54**	29.26	**42.36**	34.28	**50.29**
Belebele (MC)	Acc.	32.75	**35.95**	37.66	**43.47**	41.64	**53.77**
MMLU	Acc/char	26.74	**28.03**	27.90	**29.86**	29.30	**32.23**
ARC-Challenge Multilingual	Acc/char	25.28	**27.74**	26.66	**31.47**	29.50	**36.24**
Belebele (RC)	Acc/char	33.18	**33.26**	36.85	**38.86**	39.52	**41.65**
Long-context retrieval							
RULER 4k	Macro	69.32	**72.69**	73.70	**74.49**	**82.98**	79.10
RULER 8k	Macro	60.34	**64.06**	63.25	**67.24**	**77.01**	74.00
RULER 16k	Macro	52.20	**53.81**	58.07	**58.71**	**70.58**	67.32
RULER 32k	Macro	**49.25**	47.95	50.46	**52.17**	60.27	**62.53**
RULER 64k	Macro	**46.38**	43.68	42.60	**44.78**	**55.10**	54.57
RULER mean	Macro	55.50	**56.44**	57.62	**59.48**	**69.19**	67.50

Table 11 Context-extended model results grouped by capability. Scores are on a 0–100 scale. Bold indicates the higher score within each BASELINE–CURATED pair. RULER mean gives equal weight to the five context lengths.

		M		L		XL	
Benchmark	**Metric**	**Baseline**	**Curated**	**Baseline**	**Curated**	**Baseline**	**Curated**
Knowledge and reasoning							
AGIEval English	Macro	20.35	**24.02**	28.68	**40.15**	41.77	**50.15**
MMLU-Pro	Macro	9.45	**15.00**	14.43	**26.07**	23.47	**33.27**
GPQA Diamond	EM	8.59	**9.09**	10.61	**13.13**	**20.71**	16.67
TriviaQA short answer	EM	5.42	**7.47**	10.65	**10.72**	13.90	**13.99**
Instruction following and writing							
IFEval	Prompt loose	42.14	**48.61**	53.60	**57.49**	**64.51**	62.48
IFBench	Prompt loose	8.33	**13.00**	13.00	**13.67**	**17.67**	17.33
Arena-Hard Writing	Score	28.28	**37.00**	38.76	**42.76**	49.00	**51.76**
Mathematics							
MATH-500	Math verify	30.00	**54.60**	60.60	**73.60**	73.40	**85.80**
GSM8K	EM-flex	24.11	**39.95**	50.87	**65.05**	62.62	**76.88**
AIME 2025	pass@1	2.60	**8.75**	12.50	**24.69**	19.58	**35.10**
AIME 2024	pass@1	1.67	**6.35**	9.06	**18.85**	15.21	**41.88**
AMC23	pass@1	11.95	**26.41**	30.31	**50.39**	46.33	**71.72**
HMMT 2025	EM-flex	0.00	0.00	3.33	**6.67**	10.00	**16.67**
Code							
HumanEval++	pass@1	19.57	**34.09**	34.88	**40.06**	45.73	**64.45**
MBPP++	pass@1	26.96	**38.60**	33.02	**37.59**	42.14	**50.93**
LiveCodeBench	pass@1	4.26	**7.55**	8.20	**14.44**	12.55	**19.82**
MultiPL-E+	Macro pass@1	6.25	**17.79**	13.56	**16.80**	13.71	**23.52**
CRUXEval (input + output)	pass@1	0.02	**0.11**	0.37	**8.12**	1.71	**17.29**
Multilingual							
MMMLU	Macro	12.43	**18.23**	16.23	**20.23**	20.11	**22.00**
MGSM	Macro	2.73	**6.76**	8.47	**16.91**	16.11	**21.05**
Tool use							
BFCL v3 non-live	Acc.	49.14	**56.19**	63.60	**71.22**	67.48	**75.11**
BFCL v3 live	Acc.	35.67	**41.89**	44.69	**53.53**	51.53	**59.00**
Nexus	Acc.	19.50	**34.28**	34.91	**37.11**	38.99	**42.45**
SEAL	Param F1	65.14	**70.87**	74.13	**79.15**	78.50	**82.22**
Format check	*Acc.*	99.56	**99.70**	**99.70**	99.48	**100.00**	99.70
Tool check	*Tool F1*	**86.47**	84.43	86.14	**90.25**	90.94	**92.55**
Long context							
RULER 32k	Macro	13.81	**19.03**	20.27	**28.72**	23.13	**32.30**

Table 12 Results after supervised fine-tuning. Scores use the displayed metric on a 0–100 scale. Code results use pass@1; CRUXEval averages input and output prediction. Indented italic rows are SEAL's fail-fast format and tool checks, which gate its parameter-F1 score. Bold indicates the higher score within each pair; ties are not bolded.

Table 13 Gain of each RLVR specialist over its SFT parent, averaged over the domain's benchmarks (percentage points).

	M		L		XL	
Domain	BASELINE	CURATED	BASELINE	CURATED	BASELINE	CURATED
Math	+2.4	**+3.2**	+3.5	**+4.9**	**+7.0**	+4.7
Code	+5.0	**+7.2**	+4.2	**+8.3**	**+8.3**	+5.9
IF	**+15.1**	+9.1	**+9.5**	+7.3	**+10.0**	+8.8
Tool use	**+4.6**	+0.5	+5.4	**+6.5**	**+7.5**	+6.3

RLVR specialists. Table 13 reports the gain of each RLVR specialist over its SFT parent; Table 18 in Appendix B.3 gives the per-benchmark results.

Specialist merging. After specialist merging, the CURATED models retain an overall advantage at all three scales on the 25-benchmark mean (Table 14); § 7.5 examines how that advantage scales.

At XL scale, CURATED models outperform their BASELINE counterparts across competition mathematics, code generation, and tool use: AIME 2025 increases from 19.38 to 35.63, HumanEval++ from 54.09 to 70.18, and Nexus from 41.19 to 48.74.

Tool-use gains persist after specialist merging at all three scales, with CURATED models scoring higher on BFCL live, BFCL non-live, Nexus, and SEAL parameter F1. At XL scale, BFCL live accuracy increases from 55.18 to 62.42 and non-live accuracy from 73.67 to 80.00. SEAL parameter F1 improves from 82.08 to 85.74, and tool F1 from 92.36 to 95.09. Format accuracy remains near saturation, at 99.85 for the BASELINE model and 100.00 for the CURATED model.

The advantage is less consistent on instruction following. CURATED models lead on IFEval at all three scales, but score lower on IFBench at the L and XL scales.

7.5 Curation is worth about 6× pretraining compute

After context extension, SFT, and specialist merging, the CURATED advantage widens with scale rather than washing out (Figure 1). On the mean of the 25 benchmark metrics in Table 14, BASELINE models score 22.9, 30.5, and 37.7 at M, L, and XL scale, and CURATED models score 29.0, 39.1, and 46.8, respectively, so the gap grows from 6.1 to 8.6 to 9.1 points.

The CURATED L model already exceeds the XL BASELINE model while using about one-fifth of the pretraining compute, scoring 39.1 versus 37.7 on the 25-benchmark mean. Its lead also holds in mathematics and code (Table 14). Averaged over MATH-500, GSM8K, AIME 2024, AIME 2025, AMC23, and HMMT 2025, it scores 44.0 versus 41.3; averaged over HumanEval++, MBPP++, LiveCodeBench, MultiPL-E+, and CRUXEval, its pass@1 is 31.6 versus 28.8.

Because the XL BASELINE score falls between the CURATED M and L scores, we can estimate the pretraining compute a CURATED model needs to match it by interpolation rather than extrapolation. Let S_{target} denote the XL BASELINE score, and let (C_1, S_1) and (C_2, S_2) denote the CURATED M and L points. Interpolating score linearly in log compute gives

$$C_{\text{match}} = C_1 \left(\frac{C_2}{C_1}\right)^{\frac{S_{\text{target}} - S_1}{S_2 - S_1}}. \tag{7.1}$$

With unrounded scores $S_{\text{target}} = 37.6700$, $S_1 = 28.9712$, and $S_2 = 39.1340$, and with $C_1 = 1.47 \times 10^{21}$ and $C_2 = 7.51 \times 10^{21}$ FLOPs (Table 16), the matching budget is 5.94×10^{21} FLOPs, which is 6.28× less than the 3.73×10^{22} FLOPs used to pretrain the XL BASELINE.

		M		L		XL	
Benchmark	**Metric**	Baseline	Curated	Baseline	Curated	Baseline	Curated
Knowledge and reasoning							
AGIEval English	Macro	25.09	**29.31**	33.34	**41.81**	42.73	**48.75**
MMLU-Pro	Macro	13.08	**16.25**	16.42	**27.66**	24.79	**32.42**
GPQA Diamond	EM	**21.21**	10.61	15.66	**16.67**	18.18	**20.71**
TriviaQA short answer	EM	7.38	**8.42**	11.65	**12.12**	**14.48**	14.00
Instruction following and writing							
IFEval	Prompt loose	56.38	**60.07**	63.22	**65.62**	65.62	**67.47**
IFBench	Prompt loose	13.67	**14.33**	**16.00**	14.00	**21.67**	18.67
Arena-Hard Writing	Score	43.20	**43.96**	**46.40**	45.20	50.76	**52.88**
Mathematics							
MATH-500	Math verify	39.20	**59.40**	60.80	**77.00**	76.40	**89.60**
GSM8K	EM-flex	31.39	**47.01**	52.84	**67.40**	67.70	**78.85**
AIME 2025	pass@1	4.69	**11.98**	12.19	**23.02**	19.38	**35.63**
AIME 2024	pass@1	2.50	**5.83**	10.10	**22.19**	18.02	**39.90**
AMC23	pass@1	20.94	**30.78**	32.42	**57.73**	52.73	**73.91**
HMMT 2025	EM-flex	0.00	**3.33**	0.00	**16.67**	13.33	**26.67**
Code							
HumanEval++	pass@1	21.28	**32.68**	38.35	**54.70**	54.09	**70.18**
MBPP++	pass@1	28.52	**44.26**	36.01	**45.37**	46.01	**56.61**
LiveCodeBench	pass@1	5.20	**9.82**	10.34	**18.25**	14.48	**21.31**
MultiPL-E+	Macro pass@1	10.69	**26.36**	17.12	**21.15**	19.03	**34.22**
CRUXEval (input + output)	pass@1	**1.60**	0.88	2.55	**18.64**	10.22	**31.57**
Multilingual							
MMMLU	Macro	22.83	**24.46**	23.00	**24.20**	23.32	**26.66**
MGSM	Macro	5.89	**10.80**	12.11	**20.22**	18.84	**25.42**
Tool use							
BFCL v3 non-live	Acc.	52.45	**56.33**	70.00	**75.32**	73.67	**80.00**
BFCL v3 live	Acc.	28.12	**35.41**	47.31	**58.86**	55.18	**62.42**
Nexus	Acc.	27.36	**37.74**	34.59	**41.82**	41.19	**48.74**
SEAL	Param F1	73.25	**78.74**	79.92	**83.48**	82.08	**85.74**
Format check	*Acc.*	99.70	99.70	99.70	**100.00**	99.85	**100.00**
Tool check	*Tool F1*	86.74	**91.18**	90.97	**93.70**	92.36	**95.09**
Long context							
RULER 32k	Macro	15.85	**25.52**	20.06	**29.25**	17.85	**27.28**

Table 14 Results after specialist merging. Scores use the displayed metric on a 0–100 scale. Code results use pass@1; CRUXEval averages input and output prediction. Indented italic rows are SEAL's fail-fast format and tool checks, which gate its parameter-F1 score; they are excluded from the 25-benchmark mean, which includes every other row. Bold indicates the higher score within each pair; ties are not bolded.

8 Contributions and Acknowledgments

Core Contributors Luke Merrick and Vidhi Jain

Technical Contributors Amro Abbas, Animesh Jha, Anshuman Suri, Derek Shi, Gabor Csapo, Haakon Mongstad, Jack Urbanek, Jasper Tan, Kaleigh Mentzer, Kalp Vyas, Matthew Leavitt, Parth Doshi, Parth Suresh, Sid Joshi, Troy Dutton, and Vineeth Dorna

Leadership Ari Morcos, Bogdan Gaza, David Schwab, and Paul Burstein

Acknowledgments Anushka Nigam, Dan Darnell, Elise Clark, and Liz Gatapia

References

A. Abbas, K. Tirumala, D. Simig, S. Ganguli, and A. S. Morcos. SemDeDup: Data-efficient learning at web-scale through semantic deduplication, 2023. URL https://arxiv.org/abs/2303.09540.

M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann, et al. Phi-4 technical report, 2024. URL https://arxiv.org/abs/2412.08905.

W. U. Ahmad, A. Ficek, M. Samadi, J. Huang, V. Noroozi, S. Majumdar, and B. Ginsburg. Opencodeinstruct: A large-scale instruction tuning dataset for code llms, 2025. URL https://arxiv.org/abs/2504.04030.

A. Ahmed. Forecasting the effects of midtraining on Delphi. https://ahmeda14960.github.io/midtraining_blog/, Oct. 2026. Blog post.

A. Albalak, D. Phung, N. Lile, R. Rafailov, K. Gandhi, L. Castricato, A. Singh, C. Blagden, V. Xiang, D. Mahan, and N. Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

L. Bandarkar, D. Liang, B. Muller, M. Artetxe, S. N. Shukla, D. Husa, N. Goyal, A. Krishnan, L. Zettlemoyer, and M. Khabsa. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants, 2023. URL https://arxiv.org/abs/2308.16884.

L. Belenki, A. Agarwal, T. Shi, and K. Toutanova. Optimizing pre-training data mixtures with mixtures of data expert models, 2025. URL https://arxiv.org/abs/2502.15950.

L. Ben Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. P. Lajarín, V. Srivastav, et al. SmolLM2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

M. Böther, J. Wills, T. Robroek, S. Xu, P. Burstein, D. Zayas, C. Blakeney, S. Joshi, H. Yin, R. Adiga, H. Mongstad, L. Merrick, P. Maini, A. Morcos, M. Leavitt, A. Klimovic, and B. Gaza. Zephon: Elastic determinism for online, stateful foundation model data loading pipelines, 2026. URL https://arxiv.org/abs/2610.03087.

F. Cassano, J. Gouwar, D. Nguyen, S. Nguyen, L. Phipps-Costin, D. Pinckney, M.-H. Yee, Y. Zi, C. J. Anderson, M. Q. Feldman, et al. Multipl-e: A scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691, 2023.

F. Chen, A. Huang, N. Golowich, S. Malladi, A. Block, J. T. Ash, A. Krishnamurthy, and D. J. Foster. The coverage principle: How pre-training enables post-training, 2025a. URL https://arxiv.org/abs/2510.15020.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

M. F. Chen, M. Y. Hu, N. Lourie, K. Cho, and C. Ré. Aioli: A unified optimization framework for language model data mixing. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://arxiv.org/abs/2411.05735.

C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DatologyAI Team. BeyondWeb: Lessons from scaling synthetic data for trillion-scale pretraining, 2025a. URL https://arxiv.org/abs/2508.10975.

DatologyAI Team. Luxical: High-speed lexical-dense text embeddings, 2025b. URL https://arxiv.org/abs/2512.09015.

DatologyAI Team. Überweb: Insights from multilingual curation for a 20-trillion-token dataset, 2026. URL https://arxiv.org/abs/2602.15210.

S. Diao, Y. Yang, Y. Fu, X. Dong, D. Su, M. Kliegl, Z. Chen, P. Belcak, Y. Suhara, H. Yin, M. Patwary, Y. Lin, J. Kautz, and P. Molchanov. CLIMB: CLustering-based iterative data mixture bootstrapping for language model pre-training, 2025. URL https://arxiv.org/abs/2504.13161.

H. Ding, Z. Wang, G. Paolini, V. Kumar, A. Deoras, D. Roth, and S. Soatto. Fewer truncations improve language modeling, 2024. URL https://arxiv.org/abs/2404.10830.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019. URL https://arxiv.org/abs/1903.00161.

S. Fan, M. Pagliardini, and M. Jaggi. DoGE: Domain reweighting with generalization estimation, 2023. URL https://arxiv.org/abs/2310.15393.

A. Fernández-Hernández, C. Pérez-Corral, J. I. Mestre, M. F. Dolz, and E. S. Quintana-Ortí. Why $\beta_1 = \beta_2$ is dynamically special in Adam, 2026. URL https://arxiv.org/abs/2601.21739.

O. Filatov, J. Wang, J. Ebert, and S. Kesselheim. Optimal scaling needs optimal norm, 2025. URL https://arxiv.org/abs/2510.03871.

D. J. Foster, Z. Mhammedi, and D. Rohatgi. Is a good foundation necessary for efficient reinforcement learning? The computational role of the base model in exploration. In *Proceedings of the Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pages 2026–2142, 2025. URL https://arxiv.org/abs/2503.07453.

C. Goddard, S. Siriwardhana, M. Ehghaghi, L. Meyers, V. Karpukhin, B. Benedict, M. McQuade, and J. Solawetz. Arcee's MergeKit: A toolkit for merging large language models. In F. Dernoncourt, D. Preoţiuc-Pietro, and A. Shimorina, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.36. URL https://aclanthology.org/2024.emnlp-industry.36.

A. Gu, B. Rozière, H. Leather, A. Solar-Lezama, G. Synnaeve, and S. I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution, 2024. URL https://arxiv.org/abs/2401.03065.

S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, et al. Textbooks are all you need, 2023. URL https://arxiv.org/abs/2306.11644.

D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. K. Li, F. Luo, Y. Xiong, and W. Liang. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

X. Han, Y. Jian, X. Hu, H. Liu, Y. Wang, Q. Fan, Y. Ai, H. Huang, R. He, Z. Yang, and Q. You. InfiMM-WebMath-40B: Advancing multimodal pre-training for enhanced mathematical reasoning, 2024. URL https://arxiv.org/abs/2409.12568.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2020. URL https://arxiv.org/abs/2009.03300.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, Y. Zhang, and B. Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

S. Hu, Y. Tu, X. Han, C. He, G. Cui, X. Long, Z. Zheng, Y. Fang, Y. Huang, W. Zhao, X. Zhang, Z. L. Thai, K. Zhang, C. Wang, Y. Yao, C. Zhao, J. Zhou, J. Cai, Z. Zhai, N. Ding, C. Jia, G. Zeng, D. Li, Z. Liu, and M. Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

K. Huang, J. Guo, Z. Li, X. Ji, J. Ge, W. Li, Y. Guo, T. Cai, H. Yuan, R. Wang, Y. Wu, M. Yin, S. Tang, Y. Huang, C. Jin, X. Chen, C. Zhang, and M. Wang. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations, 2025a. URL https://arxiv.org/abs/2502.06453.

S. Huang, T. Cheng, J. K. Liu, J. Hao, L. Song, Y. Xu, J. Yang, J. Liu, C. Zhang, L. Chai, R. Yuan, Z. Zhang, J. Fu, Q. Liu, G. Zhang, Z. Wang, Y. Qi, Y. Xu, and W. Chu. Opencoder: The open cookbook for top-tier code large language models, 2025b. URL https://arxiv.org/abs/2411.04905.

B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, et al. Qwen2.5-Coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

IBM Research. Granite 4.1 llms: How they're built. https://huggingface.co/blog/ibm-granite/granite-4-1, 2026. Official technical walkthrough.

G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

P. Intellect. prime-rl, 2025. URL https://github.com/PrimeIntellect-ai/prime-rl.

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

M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

W. Kim, A. Jagmohan, and A. Vempaty. Seal: Suite for evaluating api-use of llms, 2024. URL https://arxiv.org/abs/2409.15523.

D. Kocetkov, R. Li, L. Ben Allal, J. Li, C. Mou, C. Muñoz Ferrandis, Y. Jernite, M. Mitchell, S. Hughes, T. Wolf, D. Bahdanau, L. von Werra, and H. de Vries. The Stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.15533.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations, 2017. URL https://arxiv.org/abs/1704.04683.

V. D. Lai, C. V. Nguyen, N. T. Ngo, T. Nguyen, F. Dernoncourt, R. A. Rossi, and T. H. Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback, 2023. URL https://arxiv.org/abs/2307.16039.

Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, S. W. tau Yih, D. Fried, S. Wang, and T. Yu. Ds-1000: A natural and reliable benchmark for data science code generation, 2022. URL https://arxiv.org/abs/2211.11501.

N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith, Y. Wang, P. Dasigi, and H. Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. Villanova del Moral, T. Le Scao, L. Von Werra, C. Mou, E. González Ponferrada, H. Nguyen, et al. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 35, 2022. URL https://arxiv.org/abs/2303.03915.

K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8424–8445, 2022. doi: 10.18653/v1/2022.acl-long.577. URL https://aclanthology.org/2022.acl-long.577/.

A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2206.14858.

J. Li, A. Fang, G. Smyrnis, M. Ivgi, M. Jordan, S. Y. Gadre, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024a. URL https://arxiv.org/abs/2406.11794.

J. Li, D. Guo, D. Yang, R. Xu, Y. Wu, and J. He. Codei/o: Condensing reasoning patterns via code input-output prediction, 2025. URL https://arxiv.org/abs/2502.07316.

R. Li, L. Ben Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, et al. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023a. URL https://arxiv.org/abs/2305.06161.

T. Li, W.-L. Chiang, E. Frick, L. Dunlap, T. Wu, B. Zhu, J. E. Gonzalez, and I. Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024b. URL https://arxiv.org/abs/2406.11939.

Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee. Textbooks are all you need II: phi-1.5 technical report, 2023b. URL https://arxiv.org/abs/2309.05463.

W. Liang, T. Liu, L. Wright, W. Constable, A. Gu, C.-C. Huang, I. Zhang, W. Feng, H. Huang, J. Wang, S. Purandare, G. Nadathur, and S. Idreos. Torchtitan: One-stop pytorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SFN6Wm7YBI.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Q. Lin, M. Wen, Q. Peng, G. Nie, J. Liao, J. Wang, X. Mo, J. Zhou, C. Cheng, Y. Zhao, J. Wang, and W. Zhang. Hammer: Robust function-calling for on-device language models via function masking, 2024. URL https://arxiv.org/abs/2410.04587.

J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023a. URL https://arxiv.org/abs/2305.01210.

Q. Liu, X. Zheng, N. Muennighoff, G. Zeng, L. Dou, T. Pang, J. Jiang, and M. Lin. RegMix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5f67d864aae6115374fed7beddd119e0-Abstract-Conference.html.

T. Liu, C. Xu, and J. McAuley. Repobench: Benchmarking repository-level code auto-completion systems, 2023b. URL https://arxiv.org/abs/2306.03091.

W. Liu, X. Huang, X. Zeng, X. Hao, S. Yu, D. Li, S. Wang, W. Gan, Z. Liu, Y. Yu, Z. Wang, Y. Wang, W. Ning, Y. Hou, B. Wang, C. Wu, X. Wang, Y. Liu, Y. Wang, D. Tang, D. Tu, L. Shang, X. Jiang, R. Tang, D. Lian, Q. Liu, and E. Chen. Toolace: Winning the points of llm function calling, 2025b. URL https://arxiv.org/abs/2409.00920.

Z. Liu, T. Hoang, J. Zhang, M. Zhu, T. Lan, S. Kokane, J. Tan, W. Yao, Z. Liu, Y. Feng, R. Murthy, L. Yang, S. Savarese, J. C. Niebles, H. Wang, S. Heinecke, and C. Xiong. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets, 2024. URL https://arxiv.org/abs/2406.18518.

A. Lozhkov, R. Li, L. Ben Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei, et al. StarCoder 2 and The Stack v2: The next generation, 2024. URL https://arxiv.org/abs/2402.19173.

A. Lozhkov, H. Larcher, M. Morlon, L. Ben Allal, and L. von Werra. The stack v3: The largest open code dataset, 2026. URL https://huggingface.co/datasets/HuggingFaceCode/stack-v3-train.

R. K. Mahabadi, S. Satheesh, S. Prabhumoye, M. Patwary, M. Shoeybi, and B. Catanzaro. Nemotron-CC-Math: A 133 billion-token-scale high quality math pretraining dataset, 2025. URL https://arxiv.org/abs/2508.15096.

P. Maini, S. Seto, R. Bai, D. Grangier, Y. Zhang, and N. Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 14044–14072, 2024. doi: 10.18653/v1/2024.acl-long.757. URL https://aclanthology.org/2024.acl-long.757/.

S. Malladi, S. Jelassi, D. J. Foster, J. T. Ash, and A. Krishnamurthy. TailSFT: Filtered fine-tuning improves post-training performance, 2026. URL https://arxiv.org/abs/2608.25756.

D. N. Manh, T. P. Chau, N. L. Hai, T. T. Doan, N. V. Nguyen, Q. Pham, and N. D. Q. Bui. Codemmlu: A multi-task benchmark for assessing code understanding & reasoning capabilities of codellms, 2024. URL https://arxiv.org/abs/2410.01999.

A. Meterez, P. A. Nair, D. Morwani, C. Pehlevan, and S. Kakade. Anytime pretraining: Horizon-free learning-rate schedules with weight averaging, 2026. URL https://arxiv.org/abs/2602.03702.

T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

N. Muennighoff, A. M. Rush, B. Barak, T. Le Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.16264.

T. Nguyen, C. V. Nguyen, V. D. Lai, H. Man, N. T. Ngo, F. Dernoncourt, R. A. Rossi, and T. H. Nguyen. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, 2024. URL https://arxiv.org/abs/2309.09400.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025. URL https://arxiv.org/abs/2512.20848.

NVIDIA, A. Basant, A. Khairnar, A. Paithankar, A. Khattar, A. Renduchintala, A. Malte, A. Bercovich, A. Hazare, A. Rico, A. Ficek, A. Kondratenko, A. Shaposhnikov, A. Bukharin, A. Taghibakhshi, A. Barton, A. S. Mahabaleshwarkar, A. Shen, A. Tao, A. Guan, A. Shors, A. Mandarwal, A. Mehta, A. Venkatesan, A. Sharabiani, A. Aithal, A. Poojary, A. Dattagupta, B. Buddharaju, B. Zhu, B. Simkin, B. Kartal, B. D. Rouhani, B. Chen, B. Ginsburg, B. Norick, B. Yu, B. Catanzaro, C. Wang, C. Truong, C. Mungekar, C. Patel, C. Alexiuk, C. Munley, C. Parisien, D. Su, D. Afrimi, D. Korzekwa, D. Rohrer, D. Gitman, D. Mosallanezhad, D. Narayanan, D. Rekesh, D. Yared, D. Pykhtar, D. Ahn, D. Riach, E. Long, E. Ning, E. Chung, E. Galinkin, E. Bakhturina, G. Prasad, G. Shen, H. Qian, H. Elisha, H. Sharma, H. Ross, H. Ngo, H. Sahota, H. Wang, H. C. Shin, H. Huang, I. Cunningham, I. Gitman, I. Moshkov, J. Jung, J. Kautz, J. P. Scowcroft, J. Casper, J. Zhang, J. Zeng, J. Zhang, J. Xue, J. Huang, J. Conway, J. Kamalu, J. Cohen, J. Jennings, J. V. Vialard, J. Yi, J. Parmar, K. Briski, K. Cheung, K. Luna, K. Wyss, K. Santhanam, K. Kong, K. Pawelec, K. Anik, K. Li, K. Ahmadian, L. McAfee, L. Sleiman, L. Derczynski, L. Vega, M. R. de Melo, M. N. Sreedhar, M. Chochowski, M. Cai, M. Kliegl, M. Stepniewska-Dziubinska, M. Novikov, M. Samadi, M. Price, M. Boubdir, M. Boone, M. Evans, M. Bien, M. Zawalski, M. Martinez, M. Chrzanowski, M. Shoeybi, M. Patwary, N. Dhameja, N. Assaf, N. Habibi, N. Bhatia, N. Pope, N. Tajbakhsh, N. K. Juluru, O. Rybakov, O. Hrinchuk, O. Kuchaiev, O. Olabiyi, P. Ribalta, P. Subramanian, P. Chadha, P. Molchanov, P. Dykas, P. Jin, P. Bialecki, P. Januszewski, P. Thalasta, P. Gaikwad, P. Varshney, P. Gundecha, P. Tredak, R. K. Mahabadi, R. Patel, R. El-Yaniv, R. Rajan, R. Cheruvu, R. Shahbazyan, R. Borkar, R. Gala, R. Waleffe, R. Zhang, R. J. Hewett, R. Prenger, S. Jain, S. Kriman, S. Satheesh, S. Kaji, S. Yurick, S. Muralidharan, S. Narenthiran, S. Bak, S. Sameni, S. Han, S. Ramasamy, S. Ghosh, S. T. Sreenivas, S. Thomas, S. Diao, S. Gopal, S. Prabhumoye, S. Toshniwal, S. Ding, S. Singh, S. Jain, S. Majumdar, S. Singhal, S. Alborghetti, S. N. Akter, T. Kong, T. Moon, T. Hliwiak, T. Asida, T. Wang, T. Konuk, T. Vashishth, T. Poon, U. Karpas, V. Noroozi, V. Srinivasan, V. Korthikanti, V. Fugro, V. Kalluru, V. Kurin, V. Lavrukhin, W. U. Ahmad, W. Du, W. Byeon, X. Lu, X. Dong, Y. Karnati, Y. Choi, Y. Zhang, Y. Lin, Y. Fu, Y. Suhara, Z. Dong, Z. Li, Z. Zhu, and Z. Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

OpenAI. MMMLU: Multilingual massive multitask language understanding. https://huggingface.co/datasets/openai/MMMLU, 2024.

A. Orvieto and R. M. Gower. In search of adam's secret sauce, 2025. URL https://arxiv.org/abs/2505.21829.

K. Paster, M. Dos Santos, Z. Azerbayev, and J. Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06786.

S. G. Patil, H. Mao, F. Yan, C. C.-J. Ji, V. Suresh, I. Stoica, and J. E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning*, 2025.

G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.01116.

G. Penedo, H. Kydlíček, L. Ben Allal, A. Lozhkov, M. Mitchell, C. Raffel, L. von Werra, and T. Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

G. Penedo, H. Kydlíček, V. Sabolčec, B. Messmer, N. Foroutan, A. H. Kargaran, C. Raffel, M. Jaggi, L. von Werra, and T. Wolf. FineWeb2: One pipeline to scale them all – adapting pre-training data processing to every language, 2025. URL https://arxiv.org/abs/2506.20920.

B. Peng, J. Quesnelle, H. Fan, and E. Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

V. Pyatkin, S. Malik, V. Graf, H. Ivison, S. Huang, P. Dasigi, N. Lambert, and H. Hajishirzi. Generalizing verifiable instruction following, 2025. URL https://arxiv.org/abs/2507.02833.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training Gopher, 2021. URL https://arxiv.org/abs/2112.11446.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

M. Roemmele, C. A. Bejan, and A. S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, et al. Code Llama: Open foundation models for code, 2023. URL https://arxiv.org/abs/2308.12950.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi. Socialiqa: Commonsense reasoning about social interactions, 2019. URL https://arxiv.org/abs/1904.09728.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei. Language models are multilingual chain-of-thought reasoners, 2022. URL https://arxiv.org/abs/2210.03057.

E. Shulgin, D. von Rütte, T. H. Zhang, N. Ajroldi, B. Schölkopf, and A. Orvieto. Deriving hyperparameter scaling laws via modern optimization theory, 2026. URL https://arxiv.org/abs/2603.15958.

L. Soldaini and K. Lo. peS2o (Pretraining Efficiently on S2ORC) Dataset. Technical report, Allen Institute for AI, 2023. ODC-By, https://github.com/allenai/pes2o.

L. Soldaini, R. Kinney, A. Bhagia, D. Schwenk, D. Atkinson, R. Authur, B. Bogin, K. Chandu, J. Dumas, Y. Elazar, et al. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2402.00159.

D. Su, K. Kong, Y. Lin, J. Jennings, B. Norick, M. Kliegl, M. Patwary, M. Shoeybi, and B. Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 2459–2475, 2025. doi: 10.18653/v1/2025.acl-long.123. URL https://aclanthology.org/2025.acl-long.123/.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

A. Talmor, J. Herzig, N. Lourie, and J. Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2018. URL https://arxiv.org/abs/1811.00937.

Z. Tang, X. Zhang, B. Wang, and F. Wei. Mathscale: Scaling instruction tuning for mathematical reasoning, 2024. URL https://arxiv.org/abs/2403.02884.

P. I. Team, M. Senghaas, F. Obeid, S. Jaghouar, W. Brown, J. M. Ong, D. Auras, M. Sirovatka, J. Straube, A. Baker, S. Müller, J. Mattern, M. Basra, A. Ismail, D. Scherm, C. Miller, A. Patel, S. Kirsten, M. Sieg, C. Reetz, K. Erdem, V. Weisser, and J. Hagemann. Intellect-3: Technical report, 2025. URL https://arxiv.org/abs/2512.16144.

Team Olmo, A. Ettinger, A. Bertsch, B. Kuehl, D. Graham, D. Heineman, D. Groeneveld, F. Brahman, F. Timbers, H. Ivison, J. Morrison, J. Poznanski, K. Lo, L. Soldaini, M. Jordan, M. Chen, M. Noukhovitch, N. Lambert, P. Walsh, P. Dasigi, R. Berry, S. Malik, S. Shah, S. Geng, S. Arora, S. Gupta, T. Anderson, T. Xiao, T. Murray, T. Romero, V. Graf, A. Asai, A. Bhagia, A. Wettig, A. Liu, A. Rangapur, C. Anastasiades, C. Huang, D. Schwenk, H. Trivedi, I. Magnusson, J. Lochner, J. Liu, L. J. V. Miranda, M. Sap, M. Morgan, M. Schmitz, M. Guerquin, M. Wilson, R. Huff, R. L. Bras, R. Xin, R. Shao, S. Skjonsberg, S. Z. Shen, S. S. Li, T. Wilde, V. Pyatkin, W. Merrill, Y. Chang, Y. Gu, Z. Zeng, A. Sabharwal, L. Zettlemoyer, P. W. Koh, A. Farhadi, N. A. Smith, and H. Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

C. Tian, J. Wang, Q. Zhao, K. Chen, J. Liu, Z. Liu, J. Mao, W. X. Zhao, Z. Zhang, and J. Zhou. Wsm: Decay-free learning rate schedule via checkpoint merging for llm pre-training, 2025. URL https://arxiv.org/abs/2507.17634.

K. Tirumala, D. Simig, A. Aghajanyan, and A. S. Morcos. D4: Improving LLM pretraining via document de-duplication and diversification, 2023. URL https://arxiv.org/abs/2308.12284.

Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024a. URL https://arxiv.org/abs/2406.01574.

Z. Wang, X. Li, R. Xia, and P. Liu. MathPile: A billion-token-scale pretraining corpus for math. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 2024b. URL https://arxiv.org/abs/2312.17120.

Y. Wei, Z. Wang, J. Liu, Y. Ding, and L. Zhang. Magicoder: Empowering code generation with OSS-Instruct. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2312.02120.

J. Welbl, N. F. Liu, and M. Gardner. Crowdsourcing multiple choice science questions, 2017. URL https://arxiv.org/abs/1707.06209.

K. Wen, X. Dang, K. Lyu, T. Ma, and P. Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization, 2026. URL https://arxiv.org/abs/2606.16899.

G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, 2020. URL https://arxiv.org/abs/1911.00359.

A. Wettig, A. Gupta, S. Malik, and D. Chen. QuRating: Selecting high-quality data for training language models. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.09739.

A. Wettig, K. Lo, S. Min, H. Hajishirzi, D. Chen, and L. Soldaini. Organize the web: Constructing domains enhances pre-training data curation. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.10341.

S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. Liang, Q. V. Le, T. Ma, and A. W. Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2305.10429.

S. M. Xie, S. Santurkar, T. Ma, and P. Liang. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://arxiv.org/abs/2302.03169.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–498, 2021. URL https://arxiv.org/abs/2010.11934.

A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

A. Yang et al. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

C. Yang, R. Le, Y. Xing, Z. An, Z. Chen, W. X. Zhao, Y. Song, and T. Zhang. Toolmind technical report: A large-scale, reasoning-enhanced tool-use dataset, 2025b. URL https://arxiv.org/abs/2511.15718.

J. Ye, P. Liu, T. Sun, J. Zhan, Y. Zhou, and X. Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2403.16952.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Y. Yue, Z. Chen, R. Lu, A. Zhao, Z. Wang, Y. Yue, S. Song, and G. Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

C. Zhang, G. Neubig, and X. Yue. On the interplay of pre-training, mid-training, and RL on reasoning language models, 2025. URL https://arxiv.org/abs/2512.07783.

F. Zhang, B. Chen, Y. Zhang, J. Keung, J. Liu, D. Zan, Y. Mao, J.-G. Lou, and W. Chen. Repocoder: Repository-level code completion through iterative retrieval and generation, 2023. URL https://arxiv.org/abs/2303.12570.

Y. Zhao, Y. Qu, K. Staniszewski, S. Tworkowski, W. Liu, P. Miłoś, Y. Wu, and P. Minervini. Analysing the impact of sequence composition on language model pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7897–7912, 2024. URL https://arxiv.org/abs/2402.13991.

W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364.

F. Zhou, Z. Wang, N. Ranjan, Z. Cheng, L. Tang, G. He, Z. Liu, and E. P. Xing. Megamath: Pushing the limits of open math corpora. *arXiv preprint arXiv:2504.02807*, 2025. Preprint.

J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

T. Y. Zhuo, M. C. Vu, J. Chim, H. Hu, W. Yu, R. Widyasari, I. N. B. Yusuf, H. Zhan, J. He, I. Paul, S. Brunner, C. Gong, T. Hoang, A. R. Zebaze, X. Hong, W.-D. Li, J. Kaddour, M. Xu, Z. Zhang, P. Yadav, N. Jain, A. Gu, Z. Cheng, J. Liu, Q. Liu, Z. Wang, B. Hui, N. Muennighoff, D. Lo, D. Fried, X. Du, H. de Vries, and L. V. Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2024. URL https://arxiv.org/abs/2406.15877.

Appendix

A Pretraining Details

A.1 Systems configuration

We performed all training runs at maximum sequence length 8,192 on a cluster of server nodes equipped with 8 NVIDIA H100 SXM GPUs each and connected with a fast network interconnect. We used a fork of `torchtitan` (Liang et al., 2025), enabling compilation, some FP8 quantization, and tuning fully sharded data parallelism (FSDP) to achieve respectable throughput. We used the Zephon dataloader (Böther et al., 2026). We use the FlexAttention attention backend to mask attention across document boundaries.

Table 15 gives the systems configuration used for training at each rung. Factorization is shown as nodes × local batch size × gradient accumulation. Activation checkpointing (AC) is disabled on XS and S and uses compile-time memory budgets on larger rungs. Training throughput varied across datasets and runs, but typical values are provided for reference.

Table 15 Selected systems configurations at sequence length 8,192.

Rung	GPUs	Global batch	Factorization	AC	typical tok/s/GPU
XS	8	160	$1 \times 10 \times 2$	none	270k
S	16	224	$2 \times 7 \times 2$	none	110k
M	32	320	$4 \times 10 \times 1$	budget 0.45	41k
L	64	576	$8 \times 9 \times 1$	budget 0.10	19k
XL	128	768	$16 \times 6 \times 1$	budget 0.20	8.4k

A.2 Compute accounting

Pretraining compute figures in this report use the per-token estimate built into `torchtitan` for MoE models,

$$C_{\text{tok}} = 6\, N_{\text{active}}^{\setminus\text{emb}} + 6\, n_{\text{layers}}\, n_{\text{heads}}\, (d_{qk} + d_v)\, L_{\text{seq}}, \tag{A.1}$$

where $N_{\text{active}}^{\setminus\text{emb}}$ counts active parameters excluding the input embedding, $d_{qk} = d_v = 128$, and $L_{\text{seq}} = 8{,}192$. A run's compute is C_{tok} multiplied by its token budget. The second term counts attention over the full sequence without discounting causal or document masking. At our sequence length it accounts for 52–65% of C_{tok}, so these totals are 1.5–2× those given by the common $6ND$ approximation. Table 16 lists the resulting full-horizon compute for each rung.

Table 16 Estimated pretraining compute at the full-c horizon from Eq. (A.1). The attention share is the fraction of C_{tok} from the sequence-length term.

Rung	Tokens	C_{tok}	Attention share	Compute (FLOPs)	Share of XL
XS	35.8B	1.08×10^{9}	63%	3.89×10^{19}	0.10%
S	78.0B	2.92×10^{9}	65%	2.28×10^{20}	0.61%
M	190B	7.75×10^{9}	62%	1.47×10^{21}	3.9%
L	440B	1.71×10^{10}	57%	7.51×10^{21}	20%
XL	1T	3.73×10^{10}	52%	3.73×10^{22}	100%

A.3 Hyperparameter sweeps and learning-rate settings

Small-scale sweeps over $\beta \in \{0.90, 0.925, 0.95, 0.98\}$ with varied learning rates led us to select $\beta = 0.95$, while probes of nearby values confirmed that performance was not particularly sensitive to modest variation of β. The router learning-rate scale of 0.1 also came from small-scale sweeps.

Short-horizon learning rates. Because the shorter-horizon runs reuse the full-c batch size, B_{tok} is held constant, and Eq. (3.4) gives

$$\begin{aligned} \eta_{\text{c=160}} &= \eta_{\text{full}} \sqrt{\frac{307.12}{160}} \simeq 1.385\, \eta_{\text{full}}, \\ \eta_{\text{c=80}} &= \eta_{\text{full}} \sqrt{\frac{307.12}{80}} \simeq 1.960\, \eta_{\text{full}}. \end{aligned} \tag{A.2}$$

Table 17 lists the AdamH learning rates used at each rung and horizon. The XS and S rates come from direct short-horizon sweeps, the M rates were validated by $\sqrt{2}$ perturbations, and the L and XL rates were transferred without further tuning (§ 3.3). At every rung, the rates at the three horizons are related by Eq. (A.2).

Table 17 AdamH learning-rate settings for $\beta_1 = \beta_2 = 0.95$.

Rung	Global batch	η_{full}	$\eta_{\text{c=160}}$	$\eta_{\text{c=80}}$	Evidence for selection
XS	160	0.00396	0.00549	0.00776	direct tuning
S	224	0.00333	0.00461	0.00652	direct tuning
M	320	0.00259	0.00360	0.00508	validated transfer
L	576	0.00232	0.00321	0.00454	transfer
XL	768	0.00180	0.00249	0.00352	transfer

B Post-Training Details

This section gives the settings behind the post-training recipe summarized in §§ 5.1 and 5.2.

B.1 SFT

Optimization. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, zero weight decay, and gradient clipping at 1.0. After a 200-step warmup, the learning rate reaches 5×10^{-5}, remains constant, and then follows a square-root decay over the final 20% of steps to 5% of its peak. The nominal load-balancing coefficient is 0.05, as in pretraining. Hyperball constraints are not used.

B.2 RLVR

Shared recipe. All specialists are trained with GRPO in an asynchronous PRIME-RL setup (Intellect, 2025), with separate nodes for rollout generation and policy training. Each training batch targets 512 answers. Groups in which every answer receives the same reward are filtered out, and answers cut off by the token limit receive zero reward. We use AdamW with gradient clipping at 1.0 and token-normalized loss. Each run lasts 100 steps: the learning rate warms up over the first 15, stays constant, and decays linearly to zero over the final 30. Answers are sampled at temperature 1.0 and may come from weights at most one step behind the current policy, or two steps for tool use at XL scale. The step-100 checkpoint of each specialist is carried forward. Group size, answer-length limits, and learning rate vary by domain and are listed in Table 6.

Instruction following. Prompts come from the IFEvalG family and carry one to five verifiable constraints each; we use an easier, English-only subset. The LLM judge that gates the reward assigns zero to empty, repetitive, or claim-only responses without assessing correctness.

Tool use. The model receives a question and tool definitions and responds with native function calls in a single turn. A call counts as matched only if its name, argument values, and argument types all agree with the reference; extra or missing calls lower the score, and calls are never executed. Hammer examples mask tool names so the model must rely on descriptions, and no-tool examples reward declining without a call.

B.3 Per-benchmark specialist results

Table 18 reports each RLVR specialist's score on every benchmark in its domain, together with its gain over its SFT parent. The per-domain means match Table 13. One pattern is hidden by the domain means: both M tool-use specialists almost stop abstaining when no tool applies. Their accuracy on BFCL's irrelevance categories falls from 23–43% for their SFT parents to below 2%, which accounts for their drop on BFCL live; the L and XL tool-use specialists do not show this behavior.

C Evaluation Details

C.1 Benchmarks

For base models, knowledge and reasoning evaluations include MMLU (Hendrycks et al., 2020), MMLU-Pro (Wang et al., 2024a), AGIEval (Zhong et al., 2023), and BBH (Suzgun et al., 2022), alongside question-answering, reading-comprehension, and commonsense benchmarks: TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), CommonsenseQA (Talmor et al., 2018), DROP (Dua et al., 2019), Social IQA (Sap et al., 2019), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2019), RACE (Lai et al., 2017), OpenBookQA (Mihaylov et al., 2018), Winogrande (Sakaguchi et al., 2019), BoolQ (Clark et al., 2019), and COPA (Roemmele et al., 2011). Our AGIEval results cover only the English-language tasks included in our evaluation configuration, rather than the full benchmark; the reported aggregate averages over this subset. Mathematics evaluations cover word problems, competition mathematics, and robustness to problem variations through GSM8K (Cobbe et al., 2021), GSM-Symbolic (Mirzadeh et al., 2024), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), AMC23, College Math (Tang et al., 2024), MathPerturb (Huang et al., 2025a), and DAT Word Math. Code evaluations span function generation with HumanEval++ and MBPP++ (our corrected versions of HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021); Appendix C.3), multilingual program generation with MultiPL-E+, library-heavy and data-science generation with BigCodeBench (Zhuo et al., 2024) and DS-1000 (Lai et al., 2022), execution reasoning with CRUXEval (Gu et al., 2024) (for which we average input- and output-prediction scores), code understanding with CodeMMLU (Manh et al., 2024), and repository-level completion with RepoBench (Liu et al., 2023b) and RepoEval (Zhang et al., 2023). Multilingual evaluations include MMMLU (OpenAI, 2024), multilingual ARC-Challenge (Lai et al., 2023), and Belebele (Bandarkar et al., 2023).

After context extension, we evaluate retrieval over longer inputs using RULER (Hsieh et al., 2024) and repeat the base-model evaluations to assess retention of short-context capabilities. For SFT and RLVR models, we retain coverage of knowledge, reasoning, mathematics, code, multilingual capabilities, and long-context retrieval, adding GPQA Diamond (Rein et al., 2023), AIME 2024 and 2025, HMMT 2025, LiveCodeBench (Jain et al., 2024), and MGSM (Shi et al., 2022), together with evaluations of instruction following and tool use. IFEval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2025) test adherence to explicit instructions, Arena-Hard Writing (Li et al., 2024b) assesses writing quality, and BFCL (Patil et al., 2025), Nexus, and SEAL (Kim et al., 2024) evaluate tool selection and the correctness of generated calls and arguments. The results tables report the benchmarks and metrics used at each training stage.

C.2 Protocol

Few-shot settings. Base-model evaluations use the few-shot settings specified by each benchmark configuration. These differ across tasks: GSM8K uses eight demonstrations, MATH-500 uses four, and HumanEval++ and MBPP++ use three.

Decoding. Most reasoning configurations for post-trained models use temperature 0.6, top-$p = 0.95$, and a maximum output length of 32,768 tokens.

Context-extension retention subset. During context extension, we monitor retention of short-context capabilities on nine tasks covering code (HumanEval++, MBPP++), math (GSM8K, MATH-500, AMC23), exam-style reasoning (AGIEval), and knowledge (MMLU, MMLU-Pro, TriviaQA).

C.3 Code benchmark validation

Widely used code benchmarks contain flawed prompts, incorrect reference solutions, and broken tests that penalize correct answers or reward incorrect ones, so we audited and corrected the ones we report on instead of treating them as ground truth. All fixes were made several months before any of our training runs, and were decided from the problems and tests themselves, never from model scores.

HumanEval++ and MBPP++. Starting from the EvalPlus versions of HumanEval and MBPP (Liu et al., 2023a), we reviewed all 542 problems (164 in HumanEval+, 378 in MBPP+) and corrected every case where the prompt, reference solution, or test harness was wrong or ambiguous enough that a competent programmer could reasonably write failing code. We call the results `HumanEval++` and `MBPP++`. We modified 52 problems (9 and 43, respectively), in five categories:

1. terminology mismatches between the prompt and the reference code (15 problems; e.g., "rounded to the next integer" for nearest-integer rounding)
2. missing specifications that the solution assumes (12; e.g., equal-length inputs, or earlier dictionaries taking priority on duplicate keys)
3. incorrect docstring examples (3; e.g., the `median` example in HumanEval/47 gave 15.0 instead of 8.0)
4. buggy reference solutions or test harnesses whose tests passed by coincidence (5; e.g., MBPP/806 returned the last run of uppercase letters rather than the longest)
5. trailing GeeksforGeeks/W3Resource URLs in MBPP prompts (17 URL-only edits), which carry no semantic content and could reward memorizing the linked solutions

MultiPL-E+. MultiPL-E (Cassano et al., 2023) translates HumanEval and MBPP into many programming languages, inheriting the errors above and adding translation-specific ones. We built MultiPL-E+ (47 language shards: 24 HumanEval, 23 MBPP; 12,667 rows) in three steps:

1. propagated the HumanEval++ and MBPP++ prompt fixes (8 HumanEval and 39 MBPP problems)
2. repaired residual prompt corruption and typos, such as garbled language tokens (e.g., "swiftthon")
3. audited 11 core languages (JavaScript, TypeScript, Ruby, Go, Swift, C++, Java, C#, PHP, Rust, and Shell) by writing and executing a solution for each of 5,547 problems

The audit found 175 defective rows (3.2%), mostly type-system artifacts of translating Python into statically typed languages: wrong tuple arity in Java, untyped `interface{}` slices in Go, undefined-behavior unions in C++, and integer overflow on large expected values. We fixed 172; the other three are Shell rows whose translation loses type or nesting information. In total, 846 rows (6.7%) across 121 problems were modified. Each row records whether and why it changed, and keeps the original prompt and tests.

C.4 Known evaluation artifacts

Some evaluations produce scores that reflect the evaluation setup more than model capability. Because both models in every pair are evaluated with the same prompts, decoding, and scoring, these artifacts affect BASELINE and CURATED models alike, but they limit what individual scores can tell us.

Scores at or below chance. On a multiple-choice benchmark, a model that has learned the answer format but nothing else can achieve random-guess accuracy. Scores at or below chance indicate that the model has not reliably learned the format. GPQA Diamond and MMMLU, both four-way multiple choice, are near or below 25% for every post-trained model. Among base models, MMMLU stays between 15.17 and 17.03 at every scale, and the binary-choice BoolQ falls below 50% for the CURATED M and BASELINE L models (Table 8). The BASELINE M and L models also score almost identically on several multiple-choice benchmarks (MMLU 32.96 and 32.14, ARC-Challenge 30.03 and 30.55, OpenBookQA 33.00 and 33.00), even though their BPB improves from 0.521 to 0.448 (Table 7). In this regime, scores are driven more by whether a model has learned to answer in the expected format than by the underlying capability. This is why we use BPB as the primary measure for base models (§ 2.2), and why we treat differences on these benchmarks as weak evidence.

Base-model and post-trained scores are not comparable. Base-model and post-trained evaluations use different prompts, few-shot settings, decoding, and scoring, so a lower score after post-training does not by itself indicate lost capability. For the XL BASELINE model, TriviaQA is 49.18 as a base model and 13.90 after SFT, MBPP++ is 51.03 and 42.14, and CRUXEval is 16.46 and 1.71. We therefore compare models within a training stage rather than across stages.

		M		L		XL	
Benchmark	**Metric**	**Baseline**	**Curated**	**Baseline**	**Curated**	**Baseline**	**Curated**
Mathematics specialist (6 benchmarks)							
MATH-500	Math verify	36.60 (+6.6)	**59.20** (+4.6)	67.80 (+7.2)	**77.40** (+3.8)	78.20 (+4.8)	**89.80** (+4.0)
GSM8K	EM-flex	28.28 (+4.2)	**47.76** (+7.8)	58.83 (+8.0)	**72.33** (+7.3)	69.29 (+6.7)	**82.49** (+5.6)
AIME 2025	pass@1	0.73 (−1.9)	**8.23** (−0.5)	12.40 (−0.1)	**22.92** (−1.8)	24.06 (+4.5)	**35.94** (+0.8)
AIME 2024	pass@1	0.52 (−1.1)	**5.83** (−0.5)	10.00 (+0.9)	**16.67** (−2.2)	23.65 (+8.4)	**42.29** (+0.4)
AMC23	pass@1	18.83 (+6.9)	**31.02** (+4.6)	38.36 (+8.0)	**59.38** (+9.0)	60.70 (+14.4)	**76.02** (+4.3)
HMMT 2025	EM-flex	0.00 (+0.0)	**3.33** (+3.3)	0.00 (−3.3)	**20.00** (+13.3)	13.33 (+3.3)	**30.00** (+13.3)
Domain mean	*mean gain*	+2.4	**+3.2**	+3.5	**+4.9**	**+7.0**	+4.7
Code specialist (5 benchmarks)							
HumanEval++	pass@1	26.65 (+7.1)	**40.61** (+6.5)	38.29 (+3.4)	**56.59** (+16.5)	55.00 (+9.3)	**68.05** (+3.6)
MBPP++	pass@1	35.19 (+8.2)	**45.71** (+7.1)	39.97 (+7.0)	**49.23** (+11.6)	48.86 (+6.7)	**54.87** (+3.9)
LiveCodeBench	pass@1	5.39 (+1.1)	**7.99** (+0.4)	9.97 (+1.8)	**18.19** (+3.7)	18.64 (+6.1)	**23.99** (+4.2)
MultiPL-E+	Macro pass@1	11.12 (+4.9)	**20.10** (+2.3)	16.94 (+3.4)	**18.64** (+1.8)	23.96 (+10.3)	**33.43** (+9.9)
CRUXEval	pass@1	3.59 (+3.6)	**19.58** (+19.5)	6.03 (+5.7)	**15.73** (+7.6)	10.73 (+9.0)	**24.98** (+7.7)
Domain mean	*mean gain*	+5.0	**+7.2**	+4.2	**+8.3**	**+8.3**	+5.9
Instruction-following specialist (3 benchmarks)							
IFEval	Prompt loose	61.00 (+18.9)	**64.51** (+15.9)	**69.87** (+16.3)	66.91 (+9.4)	**78.56** (+14.0)	76.16 (+13.7)
IFBench	Prompt loose	20.67 (+12.3)	**23.33** (+10.3)	23.67 (+10.7)	23.67 (+10.0)	21.67 (+4.0)	**22.33** (+5.0)
Arena-Hard Writing	Score	**42.44** (+14.2)	38.04 (+1.0)	40.40 (+1.6)	**45.28** (+2.5)	**61.08** (+12.1)	59.48 (+7.7)
Domain mean	*mean gain*	**+15.1**	+9.1	**+9.5**	+7.3	**+10.0**	+8.8
Tool-use specialist (4 benchmarks)							
BFCL v3 non-live	Acc.	53.53 (+4.4)	**56.12** (−0.1)	72.81 (+9.2)	**81.44** (+10.2)	78.49 (+11.0)	**83.74** (+8.6)
BFCL v3 live	Acc.	25.94 (−9.7)	**27.72** (−14.2)	54.15 (+9.5)	**61.00** (+7.5)	60.60 (+9.1)	**66.06** (+7.1)
Nexus	Acc.	30.19 (+10.7)	**40.25** (+6.0)	29.56 (−5.3)	**38.05** (+0.9)	43.40 (+4.4)	**46.54** (+4.1)
SEAL	Param F1	78.19 (+13.0)	**81.06** (+10.2)	82.39 (+8.3)	**86.51** (+7.4)	84.20 (+5.7)	**87.64** (+5.4)
Domain mean	*mean gain*	**+4.6**	+0.5	+5.4	**+6.5**	**+7.5**	+6.3

Table 18 Per-benchmark results of each RLVR specialist on its own domain's benchmarks, before merging. Each cell shows the specialist's score on a 0–100 scale, using the same metrics as Table 12, followed in parentheses by its gain in percentage points over its SFT parent in Table 12. Bold marks the higher specialist score within each BASELINE–CURATED pair; ties are not bolded. Domain-mean rows average the gains within each domain and match Table 13. AIME, HMMT, and AMC23 contain 30 to 40 problems, so single-benchmark gains of a few points are within run-to-run variation.