CURRICULUM LEARNING: DOES DATA ORDER MATTER?

Anonymous authors
Paper under double-blind review

ABSTRACT

Curriculum learning, training on data in a defined order rather than at random, is one of the oldest ideas in machine learning, yet its gains in language-model pretraining have been unreliable. In this work, we ask whether data order matters at all. Holding the data mix fixed, we find that the same curriculum can help or hurt depending on the corpus. Compression-ratio interleaving improves GSM8K and MATH-500 by 5–7 percentage points (pp) on mathematical web text, but on general web text reduces ARC-Easy by 3–7 pp and GSM8K by 6–8 pp. These results suggest that curriculum effectiveness depends on whether an ordering aligns with useful structure in the corpus. We therefore introduce such structure during data generation, rewriting mathematical problems under grade-specific pedagogical rubrics. When a 1.7B model sees these 10B tokens of synthetic math, presenting grades 1 through 7 in pedagogical order raises GSM8K and MATH-500 by ~ 4 pp over randomly sampling that data. Our results suggest that when natural data lacks a useful order, synthetic rewriting offers a general way to create one.

1 INTRODUCTION

Language models are usually trained on data sampled uniformly at random from a large training corpus. Yet, what a model learns from an example depends on what it has already seen, and what it retains may depend on what it saw last. Curriculum learning is an age-old idea that a deliberate order should outperform randomly ordered data (Elman, 1993; Bengio et al., 2009). In language-model pretraining, deliberate orders still do not reliably beat shuffling (Wu et al., 2020; Zhang et al., 2026). In this work, we aim to definitively answer *whether* order matters.

We first benchmark the main families of existing curriculum methods on two corpora (general web, and mathematical web), training models at 0.6B and 1.7B parameters. These methods differ in the signal they use to rank documents, which is either a simple text statistic such as length or compression ratio or a model's judgment such as perplexity or the perplexity gap between a weak and a strong model (Zhang et al., 2025; 2026). Given the ranking, we explore data schedules that turn it into a training order by either moving through the ranking once or cycling through it several times. The same method can have opposite effects across corpora. On general web text data (DCLM corpus

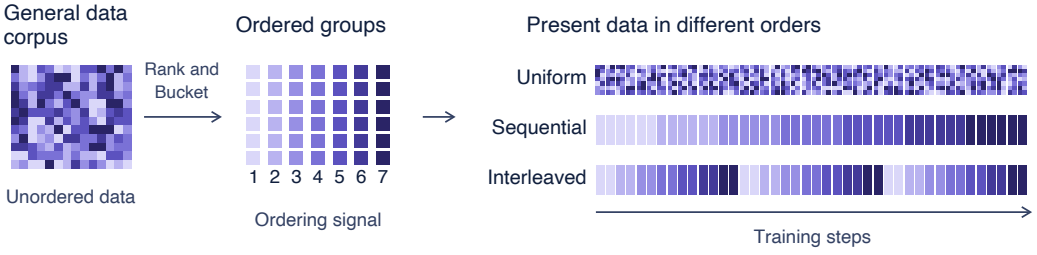


Figure 1: **A curriculum changes when a model sees each part of its training data.** We sort the data into groups, then compare a random order, one pass through the groups, and several shorter passes. Each group contributes the same amount of data in all three cases.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

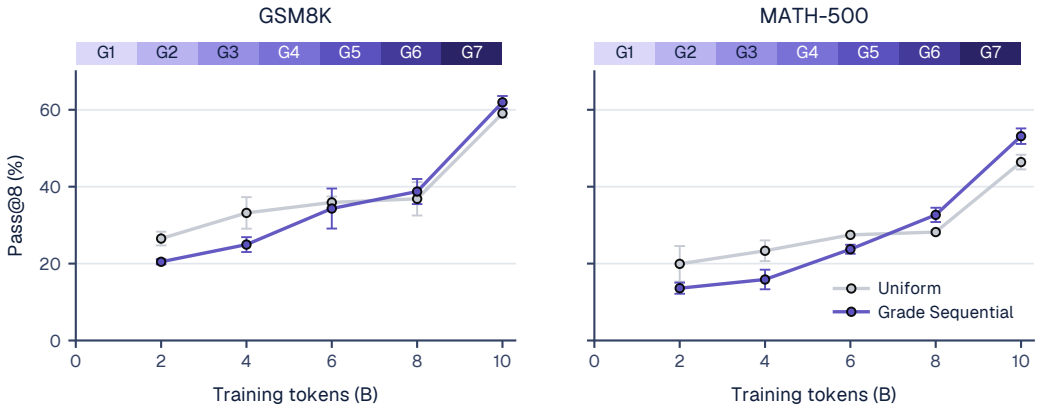


Figure 2: **Ordering data from grade 1 to grade 7 beats uniformly shuffled data.** These curves show GSM8K and MATH-500 scores during 10B-token training of Qwen3-1.7B models. Both runs see the same data once, but in a different order. The grade-ordered run starts behind because it sees earlier grades first, catches up as it reaches later grades, and finishes ahead. Curves show mean ± sample SD over three training seeds.

(Li et al., 2024)), no method beats uniformly sampled data on GSM8K at either model size, and cycling through compression-ratio based groups does the most damage, lowering ARC-Easy accuracy by 6.9 points and GSM8K pass@8 by 7.8 points at 1.7B parameters (§5.1). On our MegaMath-Web-Pro corpus, compression interleaving instead raises GSM8K and MATH-500 pass@8 by 5.3 to 7.3 points across the two model sizes (§5.2). The experiments show this reversal, but do not identify which corpus differences explain it.

The age-old idea of curriculum learning is to take inspiration from human learning, and try to instill it into model training. However, natural web corpora are not organized according to a pedagogical progression, and are often significantly biased towards certain data types. Synthetic generation gives us a way to introduce a progression explicitly. rather than trying to infer a curriculum from existing text, we can generate examples to follow a specified curriculum.

We therefore rewrite existing math problems under rubrics for grades 1 through 7, modeled on the progression of a school mathematics curriculum, so that the concepts and reasoning each problem requires are set by its grade rather than scored after the fact (Appendix A). We then train 1.7B models on 10B tokens of this corpus in either grade order or random order. Training from grade 1 through grade 7 raises MATH-500 pass@8 from 46.4% to 53.1% and GSM8K pass@8 from 59.1% to 61.9% (§5.3), and as Figure 2 shows, the grade-ordered model trails the shuffled one while it works through the early grades before pulling ahead in the later ones.

Taken together, these results suggest that curricula built on scores of existing text are unreliable when the corpus lacks an order worth following. Rewriting is already a common way to improve the quality of pretraining data (Maini et al., 2024; Su et al., 2025), and our findings suggest it can also give that data the kind of structure that human curricula have long provided.

2 Related Work

2.1 Curriculum Learning and Data Ordering

Curriculum learning replaces uniformly shuffled training with a structured progression over examples, traditionally from easier to harder data (Bengio et al., 2009). Subsequent work has shown that curriculum effectiveness depends jointly on how difficulty is estimated and how examples are scheduled during optimization (Kocmi & Bojar, 2017; Zhang et al., 2018; Hacohen & Weinshall, 2019; Platanios et al., 2019; Xu et al., 2020). Curricula can also be adapted to the learner during training, rather than specified entirely in advance (Graves et al., 2017). In language modeling, staged training

from shorter to longer contexts provides an early example of curriculum-like pretraining in which the training distribution changes systematically over time (Press et al., 2021).

Recent work has studied data ordering directly in large-scale language-model training. Zhang et al. (2026) evaluate multiple document-level signals and pacing strategies for pretraining, including compression-based orderings and interleaved curricula. Other approaches construct curricula from model-dependent signals, including perplexity differences between models (Zhang et al., 2025) and training-data influence (Schoenegger et al., 2025). More broadly, Dai et al. (2026) show that properties such as cyclic scheduling, curriculum continuity, and local diversity affect the utility of a fixed data ranking. Curriculum direction is also not universally determined: in mathematical post-training, Jia et al. (2026) find that forward and reverse orderings can differ across models, tasks, and difficulty measures. Finally, curriculum effects can depend on optimization itself; Luo et al. (2026) show that learning-rate decay can reduce the benefit of placing high-quality data late in training. Together, these studies motivate separating the ordering signal from the schedule used to realize that ordering.

2.2 Structured Skills and Pedagogically Controlled Data

A related line of work organizes training around skills, developmental structure, or explicit pedagogical constraints. Skill-It formalizes language-model training in terms of interdependent skills and studies whether relationships between skills can be exploited for data-efficient learning (Chen et al., 2023). Developmentally motivated language-modeling work has similarly investigated age-ordered or linguistically structured curricula (Opper et al., 2023; Salhan et al., 2024), while other approaches progressively control which linguistic structures or forms are exposed during training (Lucas et al., 2024; Güven et al., 2025). These studies emphasize that useful learning structure need not be captured by a single scalar measure of document difficulty.

Synthetic generation provides a complementary way to control the structure of the training distribution. TinyStories restricts generated text to vocabulary and concepts suitable for young children, while the Phi line demonstrates the use of synthetically generated "textbook-quality" material for pretraining (Eldan & Li, 2023; Gunasekar et al., 2023). More directly related to curricula, Roque & Velasco (2025) combine LLM-based text simplification with complexity-based data schedules, and CurLL constructs a large synthetic developmental corpus with explicit stages, skills, and prerequisite relationships (Tankala et al., 2025). LittleLearner similarly constructs a large pretraining corpus whose knowledge exposure is bounded by elementary-school curriculum guidelines (Li et al., 2026a). In mathematical reasoning, MathMixup generates difficulty-controlled problems and uses the resulting graded data for curriculum-based fine-tuning (Li et al., 2026b).

Our setting follows a similar data-centric view but differs by directly comparing generic offline curricula across natural and synthetically structured corpora, allowing us to study how corpus construction interacts with temporal data ordering during from-scratch pretraining.

2.3 Data Selection and Mixture Optimization

Curriculum learning is also related to methods that optimize which data a language model receives. DoReMi and DoGE use proxy optimization to learn effective proportions over pretraining domains (Xie et al., 2023; Fan et al., 2023), while RegMix predicts effective mixtures from experiments with smaller models (Liu et al., 2025). Adaptive approaches such as ADO and VelociTune instead change domain proportions during target-model training according to estimated learning potential or learning velocity (Jiang et al., 2025; Luo et al., 2025). These methods primarily address *how much* data should be drawn from different sources. Our setting is deliberately narrower: the corpus, ordering information, and schedule are specified before target-model training, allowing us to isolate the effect of *when* different parts of a fixed corpus are presented and how that effect depends on corpus structure.

3 Method

We study offline curricula specified before target-model training. Each curriculum consists of an ordering signal that partitions the dataset into ordered groups and a schedule that determines their presentation over training. The ordering and schedule are fixed before training, without adapting

to the target model's learning dynamics or downstream evaluation results. Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ denote a pretraining corpus of n documents. An ordering function $s : \mathcal{D} \to \mathbb{R}$ assigns a score to each document. Depending on the experiment, this score represents a textual property, a frozen model's assessment, or an external label; it need not measure difficulty. We rank documents by s in the chosen direction and partition them into K ordered buckets, $\mathcal{B} = (B_1, \ldots, B_K)$. Documents are shuffled within each bucket rather than presented in strict score order. This construction retains a coarse progression without imposing an exact order within each group. A curriculum schedule specifies the probability of sampling from each bucket at training step t:

$$p_t(k) = \Pr(\text{sample from } B_k \text{ at step } t), \qquad \sum_{k=1}^{K} p_t(k) = 1.$$

3.1 Ordering Signals

Following Zhang et al. (2026), we consider document length and compression ratio as model-independent signals. Length orders documents from fewer to more tokens. Compression orders them from most to least compressible, using the ratio of uncompressed to compressed text length.

Model based signals instead use reference language models to assess each document. Perplexity orders documents from lower to higher perplexity under a frozen reference model. PDPC (Zhang et al., 2025) compares weak and strong reference models using $s(x) = 1 - \mathrm{PPL}_{\mathrm{strong}}(x)/\mathrm{PPL}_{\mathrm{weak}}(x)$, progressing from lower to higher scores.

We also consider an ordering supplied during data generation: mathematical problems and solutions are generated under grade-specific pedagogical rubrics, and their assigned grades define the progression $G_1 \to \cdots \to G_7$. The rubrics shape the content of the generated examples rather than label existing documents after generation. This provides an intended educational progression, without assuming that every example in a later grade is harder than every example in an earlier grade.

3.2 Data Schedules

Given an ordered partition, a sequential schedule traverses the groups once, $B_1 \to \cdots \to B_K$, allocating a contiguous phase of training to each group. An interleaved schedule repeats this progression over shorter cycles, following Zhang et al. (2026). Under equal allocation, each group receives T/K updates in the sequential schedule and $T/(IK)$ updates in each of I interleaved cycles, where T is the total training budget. The schedules therefore preserve the overall allocation to each group while changing when it is presented.

Uniform random sampling is our primary control: it uses the same selected documents and training budget without conditioning presentation on the ordering signal. To test whether the direction of an ordering matters, we also reverse selected curricula to traverse $B_K \to \cdots \to B_1$, retaining the same groups and schedule. Uniform comparisons test whether the curriculum improves over random presentation; forward–reverse comparisons test whether its direction matters beyond grouping similar documents together in time.

3.3 Fixed-Data 10B Comparison

For the 10B-token synthetic comparison, we organize the data into 2,441,408 training sequences of 4,096 tokens before training. Every run uses each sequence exactly once. Uniform training shuffles the full collection; the grade curriculum traverses grades 1 through 7 once; and grade interleaving repeats that traversal over ten cycles. During a grade's phase, 95% of the scheduled data come from that grade and 5% from other grades, subject to the available sequences. The compression-based interleaved arm likewise cycles through seven ordered compression groups ten times. This comparison holds the training data fixed while changing their order.

4 Experimental Setup

We study training order in three settings: general web text, mathematical web text, and synthetic mathematics with an intended grade progression. The first two test whether scores computed from

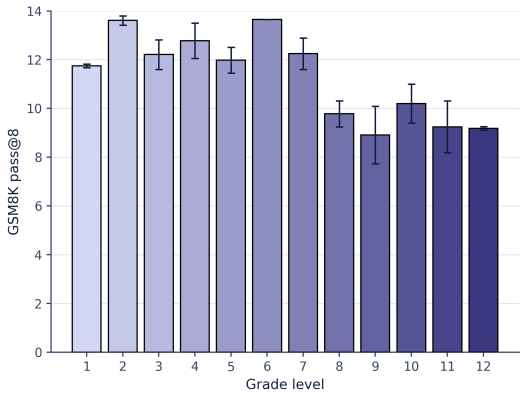


Figure 3: **Standalone performance by synthetic grade level.** We train a separate Qwen3-0.6B model for approximately 1B tokens on each grade and evaluate it on GSM8K using `pass@8`. Performance stays relatively strong across Grades 1–7, then drops for later grades. We use Grades 1–7 in the main experiments based on this ablation.

existing documents provide a useful order. The synthetic setting tests an order introduced during data generation. Within each comparison, methods use the same selected data and training budget; the curriculum changes when examples appear.

4.1 Data and Curricula

For general web text, we use DCLM (Li et al., 2024). For mathematical web text, we use our slightly refined MegaMath-Web-Pro-based corpus (Zhou et al., 2025). We select approximately one billion content tokens from each corpus before training. The same selected documents are used across methods and seeds. Appendix C describes the source corpora and selection process.

Our experiments use Grades 1–7 ($G_1, \ldots, G_7$), a range chosen through a standalone grade-level ablation. We trained a separate Qwen3-0.6B model for about 1B tokens on each grade and evaluated it on GSM8K using `pass@8`. As Figure 3 shows, performance stays relatively strong through Grade 7 and drops for later grades; models trained on Grades 8–12 score below the strongest models trained on earlier grades. We use Grades 1–7, matching the grade-school math focus of GSM8K.

Our synthetic mathematics examples are generated from source problems in OpenMathInstruct-2 (Toshniwal et al., 2025) and NuminaMath-1.5, using GPT-OSS-20B and one of seven grade-specific rubrics. For the 1BT comparison, the selected data include approximately equal content-token amounts from Grades 1–7. The 10BT comparison uses a larger selection from the same generated corpus. The generation process and rubrics are in Appendix A; data selection for both budgets is described in Appendix C.

On the two web corpora, we compare uniform sampling with curricula based on document length, compression, perplexity, and PDPC. Length and compression divide documents into ten ordered groups; perplexity uses the same number of groups, while PDPC divides documents into two groups. Depending on the method, groups are presented in one pass or revisited over ten cycles. We also test reverse order for selected curricula. The synthetic 1BT comparison applies the same score-based methods and tests both directions of the grade order, with and without interleaving. During a grade phase, 95% of examples come from the active grade and 5% from the other grades; equal phase lengths keep the overall grade proportions balanced. Appendix D gives the scoring rules and schedule details, including the adaptations used here.

The 10BT synthetic comparison isolates the effect of order: it uses the same 2,441,408 sequences of 4,096 tokens in every condition, and each sequence is used once. We compare uniform sampling, compression interleaving, sequential Grade 1-to-7 order, and grade interleaving. The two interleaved schedules make ten passes through their ordered groups. The grade schedules retain the 95% active-grade and 5% background allocation. See Section 3.3 for the data construction and Appendix D for implementation details.

Table 1: **DCLM results at the 1BT budget.** ARC-Easy and PIQA report 5-shot accuracy; GSM8K and MATH-500 report pass@8. Uniform reports absolute mean ± sample SD across three seeds; other rows report the mean difference from Uniform (pp) ± the method's SD. Color indicates the mean difference on a shared ±8 pp scale, not statistical significance. Full scores appear in Table 7.

Method	Qwen3-0.6B				Qwen3-1.7B			
	ARC-Easy	PIQA	GSM8K	MATH-500	ARC-Easy	PIQA	GSM8K	MATH-500
Uniform	46.2±0.4	61.5±0.7	12.7±0.7	10.9±0.5	49.3±0.5	63.4±0.5	12.5±0.8	9.5±0.7
	Δ mean (pp) ± method SD							
PDPC	+1.2±1.3	+1.0±0.3	−0.8±0.9	−0.8±0.6	+0.7±0.7	+0.5±0.5	−1.0±0.3	+1.2±1.1
PPL + linear	−2.6±0.7	+0.6±0.6	−1.4±0.7	−2.0±1.8	−3.0±0.8	+0.6±0.5	−1.3±0.4	−1.0±1.4
Comp.	+0.2±0.2	+0.5±0.4	−1.8±0.5	−1.0±0.4	−0.9±0.1	−0.3±0.7	−1.3±0.8	+1.3±1.1
Length + inter.	+1.8±0.8	+1.5±0.5	−1.5±0.7	−1.7±1.1	−0.4±0.3	+0.7±0.3	−0.8±0.6	−0.5±0.9
Comp. + inter.	−3.4±0.6	+1.3±0.6	−5.8±1.9	−4.0±2.6	−6.9±0.7	−1.1±0.3	−7.8±1.6	−1.5±1.7

4.2 Models and Training

We train Qwen3-0.6B and Qwen3-1.7B from random initialization in the 1BT experiments. Each run uses 8,000 updates, a sequence length of 4,096 tokens, and a global batch of 32 sequences, for 1,048,576,000 training tokens. The 10BT experiment uses Qwen3-1.7B for 76,294 updates, or 10,000,007,168 training tokens. Each condition has three training seeds; seeds vary model initialization and data shuffling, while the selected corpus and curriculum definition remain fixed. All runs use AdamW, with the remaining optimization settings listed in Appendix Table 6.

4.3 Evaluation and Reporting

We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024). For each problem, we sample eight answers. We report Avg@8, the fraction of sampled answers that are correct, and pass@8, the fraction of problems with at least one correct answer. GSM8K uses flexible answer matching; MATH-500 uses Math-Verify (Kydlíček, 2025). For DCLM, we also report five-shot accuracy on ARC-Easy and PIQA. Appendix E gives the sampling and scoring settings and metric definitions.

We use the final checkpoint from each run. The main tables report the mean and sample standard deviation across three seeds. Uniform rows show absolute scores; other rows show the mean difference from Uniform and the method's standard deviation across seeds. That standard deviation describes variation in the method's scores, not uncertainty in the difference. Table colors encode the mean difference and do not indicate statistical significance. Full absolute-score tables are in Appendix Tables 7, 8, 9, and 10.

5 Results

We compare training order in three settings. DCLM asks whether document scores help on general web text; MegaMath-Web-Pro corpus asks the same question on mathematical web text; and synthetic mathematics lets us test an order supplied during data generation. We report GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024) pass@8 when evaluating mathematical reasoning.

5.1 General Web Text

On DCLM, no method improves reliably across the reported tasks. Uniform sampling has the highest GSM8K pass@8 at both model scales. At 1.7B parameters, compression interleaving lowers ARC-Easy accuracy by 6.9 percentage points and GSM8K pass@8 by 7.8 points relative to Uniform. These results do not support a general claim that imposing an order helps on broad web text.

Table 2: **Our MegaMath-Web-Pro-based corpus at the 1BT budget.** Uniform reports absolute mean ± sample SD across three seeds; other rows report the mean difference from Uniform (pp) ± the method's SD. Color indicates the mean difference on a shared ±8 pp scale, not statistical significance. Absolute scores appear in Table 8.

Method	Qwen3-0.6B				Qwen3-1.7B			
	GSM8K		MATH-500		GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Uniform	1.7±0.1	11.9±0.3	2.2±0.1	12.5±0.7	2.0±0.1	13.0±0.6	2.6±0.2	14.3±1.0
	Δ mean (pp) ± method SD							
PDPC	+0.1±0.0	+0.1±0.5	+0.1±0.1	+0.7±1.8	0.0±0.2	+0.3±0.7	+0.2±0.2	+1.3±1.1
PPL + linear	0.0±0.0	+0.1±0.3	+0.5±0.4	+2.8±1.3	+0.1±0.3	+0.7±1.7	+0.2±0.5	+0.6±1.9
Comp.	+0.2±0.3	+0.9±1.4	+0.7±0.3	+3.4±0.9	+0.4±0.1	+1.5±0.3	+0.9±0.4	+3.3±1.2
Length + inter.	+0.7±0.2	+3.1±0.9	+1.2±0.2	+5.1±0.8	+0.8±0.3	+3.4±1.6	+1.2±0.2	+4.4±1.0
Comp. + inter.	+1.2±0.1	+5.3±0.3	+2.1±0.3	+7.3±0.3	+1.5±0.4	+6.0±2.2	+2.0±0.5	+5.9±1.4

The DCLM results leave open whether the score-based orderings are simply ineffective, or whether document scores provide a weaker signal than the task-relevant structure in mathematical text. We next keep the comparison focused on mathematical evaluation and ask whether ordering documents by length or compression changes the outcome on our MegaMath-Web-Pro-based corpus.

5.2 MegaMath-Web-Pro-Based Mathematics

On our MegaMath-Web-Pro-based corpus, length and compression interleaving have higher mean GSM8K and MATH-500 pass@8 than uniform sampling at both model scales. Compression interleaving has the highest mean in all four comparisons. Other methods give mixed results. This suggests that a useful order can sometimes be found in mathematical web text, but the result depends on both the document score and how the order is presented.

This result is specific to the inferred document order: it does not tell us what happens when the order is supplied directly. Our synthetic mathematics setting lets us test that case because each example is generated under one of seven grade-specific instructions. We compare following that grade progression with uniform sampling and with the reverse direction.

5.3 Synthetic Data

The synthetic corpus provides an order through its grade-specific generation instructions. We test whether following that intended progression helps training.

In the 1B-token table, sequential G1→G7 has a higher mean pass@8 than uniform sampling on both tasks at both model scales. The gains are $+5.18$ points on GSM8K and $+2.40$ on MATH-500 at 0.6B, and $+5.71$ and $+7.20$ at 1.7B. Reversing the grade order has a different effect on the two tasks: it remains above uniform on GSM8K but falls below uniform on MATH-500. Direction matters in this comparison, but the results do not show that every reverse schedule is harmful.

The forward and reverse conditions use the same overall share of each grade and differ in when those grades appear. Their different results show that the direction of the sequence can matter, especially on MATH-500. Because direction also changes which grades arrive late in training, this comparison alone does not establish that the model has learned grade prerequisites.

The 10B-token comparison asks whether the same training sequences, each used once, lead to different outcomes when reordered. At 1.7B parameters, grade interleaving raises mean pass@8 from 59.06 to 63.18 on GSM8K and from 46.40 to 50.33 on MATH-500. Sequential G1→G7 reaches 61.94 on GSM8K and 53.13 on MATH-500, the highest MATH-500 mean among these schedules. Compression interleaving also finishes above uniform on both tasks. All four rows have three evaluated training seeds; Table 4 reports their differences from Uniform.

Table 3: **Grade-conditioned mathematics at the 1BT budget.** Arrows indicate curriculum direction; grade rows without interleaving use sequential schedules. Uniform reports absolute mean ± sample SD across three seeds; other rows report the mean difference from Uniform (pp) ± the method's SD. Color indicates the mean difference on a shared ±8 pp scale, not statistical significance. Absolute scores appear in Table 9.

Method	Qwen3-0.6B				Qwen3-1.7B			
	GSM8K		MATH-500		GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Uniform	4.1±0.7	20.3±2.1	3.7±0.2	17.5±1.2	4.5±0.7	22.3±1.8	3.9±0.3	17.7±1.6
	Δ mean (pp) ± method SD							
PPL + linear	+0.4±0.9	+1.3±2.0	+0.5±0.9	+1.2±2.4	−0.1±0.7	−0.3±1.7	+0.5±0.5	+1.5±1.9
Comp.	+1.0±0.4	+3.4±1.7	−0.2±0.6	−1.6±2.3	+0.6±1.4	+1.6±4.2	+0.3±1.9	+0.5±5.2
Length + inter.	+0.4±1.4	+1.4±3.6	−0.2±0.1	−0.5±0.7	+1.9±1.5	+3.5±3.0	+1.3±0.1	+3.1±0.9
Inv. comp. + inter.	+0.3±1.9	+0.4±5.3	0.0±1.1	−1.0±3.9	+2.8±1.4	+5.4±3.9	+1.4±1.0	+5.7±2.9
Comp. + inter.	−0.6±2.6	−4.5±7.6	+0.1±0.7	−0.8±1.9	+2.2±0.2	+2.8±0.7	+1.1±0.3	+3.5±1.4
G7→G1 + inter.	+0.5±1.6	+0.9±4.7	+0.4±0.2	+0.7±1.7	+0.5±0.3	+0.7±0.9	+1.0±0.1	+2.7±0.7
G1→G7 + inter.	+1.3±1.3	+2.9±4.0	0.0±1.1	−0.6±2.6	+3.7±0.8	+7.2±1.9	+2.1±1.7	+7.1±3.8
G7→G1	+0.3±0.5	+1.3±1.1	−0.8±0.3	−3.1±1.7	+0.8±0.5	+1.9±1.3	−0.8±0.6	−2.5±2.5
G1→G7	+1.8±0.9	+5.2±1.9	+0.8±0.7	+2.4±2.5	+2.6±1.1	+5.7±2.7	+2.8±0.6	+7.2±0.9

Table 4: **Qwen3-1.7B on grade-conditioned mathematics at the 10BT budget.** Grade curricula follow G1→G7. Uniform reports absolute mean ± sample SD across three seeds; other rows report the mean difference from Uniform (pp) ± the method's SD. Color indicates the mean difference on a shared ±8 pp scale, not statistical significance. Absolute scores appear in Table 10.

Method	GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8
Uniform	28.6±1.2	59.1±1.1	21.1±1.0	46.4±1.9
	Δ mean (pp) ± method SD			
Comp. + inter.	+1.1±5.8	+3.5±4.0	+0.7±1.0	+2.7±1.3
G1→G7 + inter.	+2.9±5.3	+4.1±4.1	+2.7±2.0	+3.9±2.2
G1→G7	+1.5±2.4	+2.9±1.6	+3.8±2.0	+6.7±2.0

Across these settings, the results differ by corpus and by the order being tested. Score-based orders do not help consistently on DCLM, while interleaving by document score has higher means on the mathematical web corpus. In the synthetic setting, where the grade progression is known, the sequential G1→G7 schedule is above uniform in both budgets, though the reverse schedule shows that this advantage does not extend to every ordering. The same-data 10B comparison further shows that changing sequence order can change the final scores even when each sequence is used once. Together, these results point to order as a factor whose effect depends on the data and the schedule, rather than to one ordering rule that works across settings.

5.4 Does the Advantage Survive a Uniform Ending?

Could grade order help only because the model sees later grades near the end? In this run, we finish the grade progression earlier and train on the uniform mixture for the final 16,128 steps, starting at step 64,064. We call this schedule *uniform-ending*. Table 5 reports GSM8K and MATH-500 pass@8 at step 60,144, the last evaluated checkpoint before learning-rate decay, and at the final step, 80,192. The earlier checkpoint is 3,920 steps before the uniform phase begins. Scores are means over the full test set and four training seeds. Each gap is the mean of the four paired differences from Uniform.

Table 5: **A grade progression followed by uniform training.** GSM8K and MATH-500 pass@8 scores (%) are full-test means over four seeds. The gap is the mean paired difference between uniform-ending and Uniform, in percentage points; rounding the displayed means can shift a difference by 0.01 point. Forward follows the grade progression without the final uniform phase. The uniform-ending phase starts at step 64,064. This is a separate comparison from the fixed-data 10B study in Table 4.

Task	Step	Uniform	Uniform-ending	Gap (points)	G1→G7
GSM8K	60,144	33.45	35.60	+2.14	35.46
GSM8K	80,192	54.17	58.89	+4.72	58.95
MATH-500	60,144	22.70	29.10	+6.40	26.10
MATH-500	80,192	41.45	42.85	+1.40	47.80

At the final checkpoint, uniform-ending is 4.72 points above Uniform on GSM8K, close to Forward's 4.78-point gap. All four uniform-ending seeds score higher than their paired Uniform seeds. On MATH-500, the uniform-ending gap is 1.40 points, compared with Forward's 6.35; the seed-based interval for uniform-ending spans zero. The earlier GSM8K gap of 2.14 points depends heavily on one seed, whose paired difference is 10.69 points. Two of the other three differences are negative. The grade phases are shorter in this run, and most of the uniform phase overlaps learning-rate decay, so the comparison cannot isolate the effect of ending with the uniform mixture. It does show that the GSM8K advantage remains after substantial uniform training, while the MATH-500 advantage is smaller at the final checkpoint.

6 Discussion and Conclusion

Changing data order does not help equally in every setting. On general web text, none of the orders we tested improves both reported tasks over uniform sampling. On mathematical web text, length and compression interleaving raise the mean scores in the comparisons reported here. The clearest test is the 10B-token synthetic study: each schedule presents the same training sequences exactly once, yet grade interleaving improves mean pass@8 by 4.12 points on GSM8K and 3.93 points on MATH-500 over uniform sampling at 1.7B parameters. The benefit therefore cannot be attributed to a different overall data mix.

One possible reason interleaving helps is that it reduces the need to choose the perfect time to leave a group. If a schedule moves on too early, it can present the group's remaining data in a later cycle; if it stays too long, the model may have to adapt sharply when the next group arrives. This is a hypothesis, not a mechanism established by our results. Changing the number and length of cycles while tracking what models learn and forget would help test it.

One might wonder whether G1→G7 helps because grade 7 happens to come last. We probe a related question by finishing the grade sequence earlier, then returning to a uniform mixture for the final 16,128 steps (§ 5.4). The GSM8K advantage remains $+4.72$ points over Uniform, close to $+4.78$ for the run that ends on later grades. On MATH-500, however, the advantage narrows to $+1.40$ points, compared with $+6.35$ for the grade-ordered run. The GSM8K gain therefore survives a return to mixed-grade training, but the MATH-500 gain does not persist to the same extent. Because the grade phases also become shorter, this experiment does not isolate recency from the rest of the schedule.

The same curriculum can raise scores on one corpus and lower them on another, so an ordering only has meaning in relation to the data it arranges. Our fixed-data synthetic comparison also shows that changing when examples appear can change final performance even when every training sequence is held constant. Generating examples under grade-specific instructions gives us one way to introduce a progression before training begins. This makes generation useful not only for creating examples, but also for shaping how those examples relate to one another.

AI USE STATEMENT

The authors conceived the research questions and central ideas, made the experimental decisions, and own the scientific conclusions. We used generative AI extensively to accelerate the work. A language model generated the grade-conditioned mathematical problems and solutions studied here, as described in §A. AI assistants helped implement and debug experimental code, check and analyze results, give feedback on experimental plans, and prepare figures. They also helped turn the authors' notes, ideas, and experimental details into manuscript prose and polish that prose. The authors reviewed and revised AI-assisted work and take responsibility for the experiments, reported results, and every claim in the final paper. Even where AI helped with the wording, the paper expresses the authors' scientific judgment and voice.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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

Yalun Dai, Yangyu Huang, Tongshen Yang, Yonghan Wang, Xin Zhang, Wenshan Wu, Qihao Zhao, Hao Li, Yuanyuan Gao, Kim-Hui Yap, and Scarlett Li. Demystifying data organization for enhanced llm training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 27358–27375, 2026. doi: 10.18653/v1/2026.acl-long.1262.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993. doi: 10.1016/0010-0277(93)90058-4.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393*, 2023.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1311–1320, 2017.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Arzu Burcu Güven, Anna Rogers, and Rob van der Goot. Do syntactic categories help in developmentally motivated curriculum learning for language models? In *Proceedings of the First BabyLM Workshop*, pp. 288–300, 2025. doi: 10.18653/v1/2025.babylm-main.22.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2535–2544, 2019.

Yaning Jia, Chunhui Zhang, Xingjian Diao, Xiangchi Yuan, Zhongyu Ouyang, Chiyu Ma, and Soroush Vosoughi. What makes a good curriculum? disentangling the effects of data ordering on llm mathematical reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34472–34488, 2026.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. In *International Conference on Learning Representations*, volume 2025, pp. 58014–58037, 2025.

Tom Kocmi and Ondřej Bojar. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pp. 379–386, 2017. doi: 10.26615/978-954-452-049-6_050.

Hynek Kydlíček. Math-Verify: Math verification library, 2025. URL https://github.com/huggingface/math-verify.

Fanfei Li, Jana Zeller, Manuel Prada-Corral, Thaddäus Wiedemer, Prasanna Mayilvahanan, Ryan Cotterell, and Wieland Brendel. LittleLearner: Language models under pedagogically-controlled knowledge exposure, 2026a.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14200–14282, 2024.

Xuchen Li, Jing Chen, Xuzhao Li, Hao Liang, Xiaohuan Zhou, Taifeng Wang, and Wentao Zhang. Mathmixup: Boosting llm mathematical reasoning with difficulty-controllable data synthesis and curriculum learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2497–2513, 2026b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. In *International Conference on Learning Representations*, volume 2025, pp. 38305–38339, 2025.

Evan Lucas, Dylan Gaines, Tagore Rao Kosireddy, Kevin Li, and Timothy C. Havens. Using curriculum masking based on child language development to train a large language model with limited training data. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pp. 221–228, 2024.

Kairong Luo, Zhenbo Sun, Haodong Wen, Xinyu Shi, Jiarui Cui, Chenyi Dang, Kaifeng Lyu, and Wenguang Chen. How learning rate decay wastes your best data in curriculum-based llm pretraining. In *International Conference on Learning Representations*, volume 2026, pp. 147266–147297, 2026.

Zheheng Luo, Xin Zhang, Xiao Liu, Haoling Li, Yeyun Gong, Qi Chen, and Peng Cheng. Velocitune: A velocity-based dynamic domain reweighting method for continual pre-training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16644–16656, 2025.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14044–14072, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.757. URL https://aclanthology.org/2024.acl-long.757/.

Mattia Opper, James Morrison, and N Siddharth. On the effect of curriculum learning with developmental data for grammar acquisition. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pp. 346–355, 2023.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 1162–1172, 2019.

Ofir Press, Noah A Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5493–5505, 2021.

Matthew Theodore Roque and Dan John Velasco. Beyond repetition: Text simplification and curriculum learning for data-constrained pretraining. In *Proceedings of the First BabyLM Workshop*, pp. 246–255, 2025. doi: 10.18653/v1/2025.babylm-main.19.

Suchir Salhan, Richard Diehl Martinez, Zébulon Goriely, and Paula Buttery. Less is more: Pretraining cross-lingual small-scale language models with cognitively-plausible curriculum learning strategies. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pp. 174–188, 2024.

Loris Schoenegger, Lukas Thoma, Terra Blevins, and Benjamin Roth. Influence-driven curriculum learning for pre-training on limited data. In *Proceedings of the First BabyLM Workshop*, pp. 356–379, 2025. doi: 10.18653/v1/2025.babylm-main.26.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming common crawl into a refined long-horizon pretraining dataset. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.123. URL https://aclanthology.org/2025.acl-long.123/.

Pavan Kalyan Tankala, Shubhra Mishra, Satya Lokam, and Navin Goyal. CurLL: A developmental framework to evaluate continual learning in language models. In *Proceedings of the First BabyLM Workshop*, pp. 256–278, 2025. doi: 10.18653/v1/2025.babylm-main.20.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. In *International Conference on Learning Representations*, volume 2025, pp. 19243–19275, 2025.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? *arXiv preprint arXiv:2012.03107*, 2020.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6095–6104, 2020. doi: 10.18653/v1/2020.acl-main.542.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J. Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint arXiv:1811.00739*, 2018.

Xuemiao Zhang, Liangyu Xu, Feiyu Duan, Yongwei Zhou, Sirui Wang, Rongxiang Weng, Jingang Wang, and Xunliang Cai. Preference curriculum: Llms should always be pretrained on their preferred data. *arXiv preprint arXiv:2501.13126*, 2025.

Yang Zhang, Amr Mohamed, Hadi Abdine, Guokan Shang, and Michalis Vazirgiannis. Beyond random sampling: Efficient language model pretraining via curriculum learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5776–5794, 2026.

Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P. Xing. MegaMath: Pushing the limits of open math corpora. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=SHB0sLrZrh.

A SYNTHETIC DATASET CONSTRUCTION

We build the synthetic mathematics corpus by generating new problems and solutions from existing math examples, with a target grade specified in the prompt. The grade is part of generation; we do not assign grades to an existing set of examples afterward.

A.1 SOURCE CORPORA

We draw seed examples from two math corpora, OpenMath2 and NuminaMath-1.5. They differ in what the generator sees.

OpenMath2. We use the question-only MATH subset of OpenMathInstruct-2, released as `ai2-adapt-dev/openmath-2-math` with 11,402,286 rows. For each example, we keep the first user turn. Each seed is stored as

```
Question: <question>
```

The generator receives the source question but not its reference answer or solution. For example, a row may contain

```
Question: Find the sum of all three-digit positive integers
whose cubes end with the digits 125.
```

NuminaMath-1.5. NuminaMath-1.5 contains 863,787 rows. For this corpus, we use the full `text` field as the seed. It contains a math problem followed by its reference solution, so the generator sees both.

Here is an example row from NuminaMath-1.5:

```
Question: Solve for x if 16^x · 16^x · 16^x · 16^x = 256^10.
The equation 16^x · 16^x · 16^x · 16^x = 256^10 simplifies to
16^{4x} = 256^10.
Since 256 = 16^2, we can rewrite 256^10 as (16^2)^10, which
simplifies further to 16^20.
The equation now is 16^{4x} = 16^20. From the law of exponents,
since
the bases are equal, the exponents must be equal too:
4x = 20
x = 20/4
x = \boxed{5}
```

OpenMath2 provides only a problem statement; NuminaMath-1.5 provides a problem and its worked solution. We pass the full stored seed from each corpus to the same grade-conditioned generation procedure, described next.

A.2 GRADE-CONDITIONED CONSTRUCTION

We generate examples with `openai/gpt-oss-20b`. For each source example x_i and target grade $G_g \in \mathcal{G} = G_1, \ldots, G_7$, the model receives the source seed and instructions for that grade. It then produces a *new* math question and a complete solution. The source offers a starting point, and the grade instructions guide the new example. We denote the output by $y_{i,g}$.

We write this as $y_{i,g} \sim p_\theta(y \mid x_i, R_g)$, where θ is the generator and R_g is the rubric for grade g. Keeping x_i fixed and changing R_g gives a new version of the example for a different grade. Because the grade rubric is supplied before generation, it determines the example's grade from the start.

Each grade rubric has three fields: a `description`, a `question_instruction`, and an `answer_instruction`. For a target grade g, these fields are inserted into the following instruction template:

```
You are a mathematics education expert. Create a new
mathematical question at the
{grade} difficulty level, adapting concepts from the
original question.

Level Description: {description}
Task: {question instruction}
Solution Requirements: {answer instruction}
```

The source seed is supplied separately in the user message:

```
Original Question: {seed text}
```

The prompt includes the rubric before generation. Its three fields guide different parts of the output.

The `description` lists the math topics and content for the target grade, such as number ranges, operations, and algebra or geometry concepts. The `question_instruction` says what kind of problem to create. The `answer_instruction` says how detailed the solution should be and how it should be presented.

A.3 GRADE RUBRICS

Each grade condition is specified by three fields: a description of the mathematical content associated with the level, an instruction defining the type of question to generate, and requirements for the generated solution. Here, we share the descriptions used for Grades 1-7 verbatim below.

Grade 1

Elementary operations with numbers 1-20: single-digit addition and subtraction, counting objects, understanding place value (tens and ones), comparing numbers using $<$, $>$, $=$. Basic measurement (length, time to hour/half-hour). Simple shapes and their attributes. Problems use concrete, everyday contexts (toys, fruits, animals).

Grade 2

Two-digit operations up to 100: addition and subtraction with regrouping, skip counting by 5s/10s/100s, understanding place value to hundreds. Introduction to multiplication as repeated addition (arrays with up to 5 rows and 5 columns), recognizing even/odd numbers. Measurement: length in standard units, telling time, working with money. Simple data representation.

Grade 3

Multiplication and division foundations: multiplication/division within 100 (know all products of single-digit numbers), understanding multiplication as equal groups and division as partitioning. Introduction to fractions: unit fractions and fractions with denominators 2, 3, 4, 6, 8 using area and length models. Basic geometry: perimeter, area concepts, understanding quadrilaterals. Rounding to nearest 10 or 100.

Grade 4

Multi-digit operations: multiplication of up to 4-digit by 1-digit numbers and 2-digit by 2-digit numbers, long division. Fractions: equivalent fractions, comparing/ordering fractions with denominators 2, 3, 4, 5, 6, 8, 10, 12, 100, adding/subtracting fractions with like denominators, multiplying fractions by whole numbers. Decimals: understanding tenths and hundredths, comparing decimals. Measurement: unit conversions, perimeter, area. Factors and multiples, prime and composite numbers. Basic angle measurement.

Grade 5

Fraction operations: adding/subtracting fractions with unlike denominators, multiplying fractions (including by fractions), dividing unit fractions by whole numbers and vice versa. Decimal operations: all operations with decimals to hundredths. Place value to millions and thousandths. Volume of rectangular prisms ($V = l \times w \times h$). Coordinate plane graphing. Order of operations with parentheses. Multiplying multi-digit whole numbers, dividing by 2-digit divisors.

Grade 6

Ratios and proportional relationships: understanding ratios, rates, unit rates, percent. Division of fractions by fractions. Integer operations: all four operations with positive and negative numbers, absolute value, rational numbers on the number line, coordinate plane (all four quadrants). Introduction to expressions and equations: variable expressions, one-step equations and inequalities, dependent and independent variables. Area, surface area, volume. Greatest common factor, least common multiple. Statistical measures: mean, median, mode, range. Data displays.

Grade 7

Proportional relationships: recognizing and representing proportional relationships, solving percent problems (tax, tips, discounts, simple interest), scale drawings and scale factors. Operations with rational numbers: all operations with positive and negative fractions and decimals. Multi-step linear equations: solving equations in form p(x + q) = r, writing and solving linear inequalities. Probability: theoretical and experimental probability, compound events, probability models. Geometry: angle relationships (vertical, complementary, supplementary, angles formed by transversals), area and circumference of circles, surface area and volume of prisms, pyramids, cylinders. Sampling and inference.

A.4 Generation Configuration Details

We combine the grade instructions and source seed using the run-time chat template and tokenizer for `openai/gpt-oss-20b`. The grade instructions go in the model's system-side `model_identity` field, and the source example is the user message. We enable generation prompting and set reasoning effort to `medium`.

We tokenize each rendered prompt before generation and truncate inputs longer than 4,096 tokens to that limit. We serve `openai/gpt-oss-20b` with vLLM and MXFP4 quantization. Sampling uses temperature $= 0.8$, top-$p = 1.0$, and top-$k = 128$. We set the maximum completion length to `max_tokens` $= 8192$ and the maximum model length to `max_model_len` $= 16384$. After generation, we keep only the text after the model's `assistantfinal` marker and discard the analysis channel. The stored example is the final response, containing the new question and its solution.

B Training Hyperparameters

We describe the training hyperparameters in Table 6.

Table 6: **Training hyperparameters.** Target models start from random initialization. Primary comparisons use the final checkpoint from each run.

Setting	1BT	10BT
Reported model sizes	0.6B, 1.7B	1.7B
Context length	4,096	4,096
Global batch (sequences)	32	32
Optimizer updates	8,000	76,294
Training positions	1,048,576,000	10,000,007,168
Optimizer	AdamW	AdamW
Peak learning rate	10^{-3}	10^{-3}
Weight decay	0.1	0.1
Gradient clipping norm	1.0	1.0
LR warmup (updates)	100	100
LR decay	Final 20%, cosine	Final 20%, cosine
Final learning rate	10^{-4}	10^{-4}
Training seeds	0, 1, 2	0, 1, 2
Hardware per run	Eight H100 GPUs	Eight H100 GPUs

C Training Data

C.1 Existing Corpora

We use DCLM (Li et al., 2024) for general web text and our slightly refined MegaMath-Web-Pro-based corpus (Zhou et al., 2025) for mathematical web text. From each corpus, we select about one billion content tokens before applying any curriculum. Every method, model size, and training seed uses the same selected documents. Curriculum scores change the order of those documents, not which documents are selected.

C.2 Synthetic Training Subsets

Appendix A describes how we create the synthetic corpus. For the 1BT comparison, we select enough documents from each of Grades 1–7 to provide about one seventh of the content-token budget. All curricula, model sizes, and training seeds use the same selected subset. The 10BT comparison uses a larger shared subset from the same corpus.

Before training, we decontaminated the training data using 10-gram overlap with the evaluation data.

D Baseline Implementations

This section gives the scoring and training procedures for the baselines cited in the main text. We note where our implementation differs from the cited method.

D.1 Document Statistics and Perplexity

For the 1BT experiments, we calculate length, compression, and perplexity scores before target-model training. We rank the documents by each score and split them into ten groups of roughly equal size. We use the same scores and group assignments for every training seed.

We measure length with the Qwen tokenizer and order documents from shortest to longest. For compression, we use

$$s_{\mathrm{comp}}(x) = \frac{|\mathrm{UTF8}(x)|}{|\mathrm{zlib}_6(\mathrm{UTF8}(x))|},$$

where $|\cdot|$ denotes byte length and the subscript indicates compression level 6. We order documents from highest to lowest s_{comp}, or from most to least compressible. The reverse-compression control uses the opposite order.

We score documents with a frozen reference language model and order them from lower to higher perplexity.

For length and compression interleaving, training goes through the ten groups ten times, spending the same amount of training time on each group in every pass. The perplexity baseline goes through the groups once, again spending equal time on each. During a phase, examples come from the group being used in that phase.

The compression-to-uniform baseline goes through the compression groups ten times during the first half of training, then samples uniformly during the second half. This fixed halfway switch adapts the curriculum-warmup-then-random-sampling method of Zhang et al. (2026).

D.2 Preference Curriculum

PDPC (Zhang et al., 2025) scores each document using frozen weak and strong reference models:

$$s_{\text{PDPC}}(x) = 1 - \frac{\text{PPL}_{\text{strong}}(x)}{\text{PPL}_{\text{weak}}(x)}.$$

We use 160M- and 375M-parameter reference models and keep negative scores. We split the ranked documents into two groups of roughly equal size. At update t, the probability of drawing from the lower-scoring group is

$$p_{\text{low}}(t) = \frac{1}{1 + \exp(10(t/T - 1/2))},$$

where T is the total number of updates. The higher-scoring group is sampled with probability $1 - p_{\text{low}}(t)$.

We evaluate PDPC on DCLM and our slightly refined MegaMath-Web-Pro-based corpus. For each corpus, we use its own reference models and document scores. The reference models for the MegaMath-Web-Pro-based corpus do not score DCLM documents.

D.3 Grade Schedules

For grade-based curricula, the seven generation labels define the order. Sequential training goes from Grade 1 to Grade 7 once; interleaved training repeats that progression ten times. In the 1BT experiments, each phase draws 95% of examples from the active grade and splits the remaining 5% equally among the other six grades. Every grade gets the same amount of time as the active grade. Reverse-grade conditions use the opposite grade order with the same phase lengths and mixture.

D.4 The 10BT Comparison

The 10BT experiments compare uniform sampling, compression interleaving, sequential grade order, and grade interleaving. Both interleaved schedules repeat their groups ten times. Grade schedules go from Grade 1 to Grade 7 and aim to draw 95% of examples from the active grade and 5% from the others.

For compression interleaving, we calculate the ratio of each sequence's UTF-8 byte length to its length after zlib level-6 compression. We divide the sequences into seven ordered groups and use that order over ten cycles. This applies the document-level score to the training sequences. Grade schedules allocate the 5% background share subject to finite capacity, so every sequence is consumed once rather than reused.

D.5 Randomness

We calculate ordering scores once and reuse them across target training seeds. The training seed shuffles examples within each group and also sets the target model's initialization. Changing the seed therefore does not change the selected document pool.

Table 7: **Full results on DCLM.** Final-checkpoint scores (%) are mean ± sample SD across three training seeds.

Method	Qwen3-0.6B				Qwen3-1.7B			
	ARC-Easy	PIQA	GSM8K	MATH-500	ARC-Easy	PIQA	GSM8K	MATH-500
Uniform	46.24±0.35	61.52±0.69	12.66±0.72	10.87±0.46	49.33±0.48	63.42±0.48	12.51±0.80	9.47±0.70
PDPC	47.42±1.31	62.48±0.33	11.90±0.92	10.07±0.61	50.06±0.71	63.96±0.47	11.47±0.27	10.67±1.10
PPL + linear	43.69±0.68	62.15±0.61	11.25±0.72	8.87±1.81	46.31±0.83	64.06±0.54	11.25±0.39	8.47±1.40
Comp.	46.44±0.18	61.99±0.41	10.89±0.52	9.87±0.42	48.44±0.08	63.17±0.71	11.25±0.80	10.73±1.10
Length + inter.	48.04±0.75	63.02±0.49	11.17±0.75	9.20±1.06	48.88±0.30	64.07±0.30	11.73±0.61	8.93±0.95
Comp. + inter.	42.89±0.56	62.86±0.56	6.90±1.91	6.87±2.57	42.41±0.73	62.33±0.30	4.73±1.65	8.00±1.71

In the 10BT comparison, every run uses the same 2,441,408 sequences of 4,096 tokens, each once. The comparison therefore changes the sequence order while holding the sequence multiset fixed.

E Evaluation Protocol

E.1 Tasks and Generation

We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024). For each problem, we generate eight answers using temperature 0.8, top-p 0.95, and a maximum of 1,024 generated tokens per continuation.

We score GSM8K answers with flexible answer matching and MATH-500 answers with Math-Verify (Kydlíček, 2025), version 0.9.0. For DCLM, we also report 5-shot ARC-Easy and PIQA accuracy. These scores use answer-text likelihoods normalized by character count.

E.2 Mathematical Reasoning Metrics

For a benchmark with N problems, let $c_{ij} \in \{0, 1\}$ be 1 if the grader accepts answer j to problem i, and 0 otherwise. We sample eight answers per problem and calculate

$$\text{Avg@8} = \frac{1}{8N} \sum_{i=1}^{N} \sum_{j=1}^{8} c_{ij}, \qquad \text{pass@8} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{ \sum_{j=1}^{8} c_{ij} > 0 \right\}.$$

We multiply scores by 100 to report percentages. Avg@8 is the fraction of sampled answers accepted by the grader; pass@8 is the fraction of problems with at least one accepted answer. We calculate both metrics separately for each training seed.

E.3 Aggregation and Checkpoint Selection

Our primary comparisons use the final checkpoint from each run. For scores m_1, m_2, m_3 from the three seeds, we report

$$\bar{m} = \frac{1}{3} \sum_{r=1}^{3} m_r, \qquad s_m = \sqrt{\frac{1}{2} \sum_{r=1}^{3} (m_r - \bar{m})^2}.$$

Uniform rows show $\bar{m} \pm s_m$. Other rows in the main tables show the method's mean minus the uniform mean, alongside s_{method}. This is the method's variation across training seeds; it is not the standard deviation or a confidence interval for the difference. Table colors show the mean difference on a shared scale from -8 to $+8$ percentage points. They do not show statistical significance. The full tables give absolute scores for every method.

F Full Results tables

Table 8: **Ordering signals and schedules on our MegaMath-Web-Pro-based corpus.** Final-checkpoint scores (%) are mean ± sample SD across three training seeds.

Method	Qwen3-0.6B				Qwen3-1.7B			
	GSM8K		MATH-500		GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Uniform	1.74±0.10	11.90±0.26	2.23±0.10	12.53±0.70	1.99±0.13	12.96±0.62	2.62±0.20	14.27±1.03
PDPC	1.79±0.02	12.00±0.53	2.29±0.14	13.20±1.83	2.02±0.16	13.24±0.65	2.80±0.24	15.53±1.10
PPL + linear	1.78±0.02	12.03±0.32	2.70±0.37	15.33±1.33	2.07±0.29	13.65±1.75	2.83±0.48	14.87±1.86
Comp.	1.90±0.29	12.81±1.45	2.91±0.25	15.93±0.95	2.39±0.08	14.51±0.27	3.52±0.36	17.60±1.22
Length + inter.	2.41±0.18	15.04±0.91	3.48±0.19	17.67±0.76	2.78±0.35	16.40±1.56	3.82±0.16	18.67±1.03
Comp. + inter.	2.96±0.05	17.16±0.32	4.32±0.32	19.80±0.35	3.50±0.41	18.93±2.25	4.65±0.52	20.20±1.44

Table 9: **Ordering signals and schedules on the grade-conditioned G1–G7 corpus (1B-token setting).** Final-checkpoint scores (%) are mean ± sample SD across three training seeds.

Method	Qwen3-0.6B				Qwen3-1.7B			
	GSM8K		MATH-500		GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Uniform	4.10±0.71	20.34±2.06	3.71±0.20	17.53±1.17	4.53±0.73	22.34±1.79	3.90±0.26	17.67±1.60
PPL + linear	4.47±0.86	21.63±2.02	4.26±0.94	18.73±2.42	4.40±0.67	22.09±1.74	4.42±0.46	19.20±1.91
Comp. + warmup	5.11±0.36	23.70±1.72	3.48±0.55	15.93±2.34	5.14±1.36	23.93±4.22	4.21±1.87	18.13±5.22
Length + inter.	4.50±1.36	21.78±3.58	3.54±0.14	17.00±0.72	6.43±1.51	25.85±3.04	5.18±0.13	20.73±0.95
Inv. comp. + inter.	4.43±1.87	20.72±5.26	3.69±1.12	16.53±3.93	7.37±1.45	27.70±3.94	5.35±0.98	23.40±2.88
Comp. + inter.	3.53±2.59	15.85±7.59	3.82±0.71	16.73±1.89	6.71±0.24	25.15±0.66	4.97±0.31	21.13±1.45
G7→G1 + inter.	4.59±1.60	21.28±4.66	4.10±0.18	18.27±1.70	4.98±0.30	23.07±0.91	4.89±0.14	20.33±0.70
G1→G7 + inter.	5.38±1.34	23.20±3.95	3.71±1.05	16.93±2.57	8.22±0.76	29.52±1.87	6.01±1.66	24.73±3.83
G7→G1	4.36±0.51	21.66±1.07	2.87±0.25	14.47±1.68	5.34±0.49	24.24±1.35	3.12±0.65	15.13±2.50
G1→G7	5.92±0.88	25.52±1.94	4.54±0.69	19.93±2.55	7.17±1.12	28.05±2.71	6.67±0.58	24.87±0.90

Table 10: **Full scores for Qwen3-1.7B on synthetic grade data at 10B training tokens.** Final-checkpoint scores (%) are mean ± sample SD across three training seeds.

Method	GSM8K		MATH-500	
	avg@8	pass@8	avg@8	pass@8
Uniform	28.60±1.21	59.06±1.09	21.11±1.00	46.40±1.91
Comp. + inter.	29.66±5.77	62.55±4.03	21.82±1.01	49.13±1.27
G1→G7 + inter.	31.45±5.28	63.18±4.08	23.82±2.04	50.33±2.25
G1→G7	30.10±2.42	61.94±1.65	24.94±1.95	53.13±2.02