# Demystifying Data Organization for Enhanced LLM Training

Anonymous ACL submission


## Abstract

Large Language Models (LLMs) have revolutionized various fields, yet their training efficiency is heavily reliant on effective data curation. While data selection has been widely studied, the strategic data organization for enhanced training remains an underexplored area, particularly since current LLMs are often trained for only one or a few epochs. This paper systematically explores the influence of data organization on LLM training by reusing pre-computed sample-level scores originally generated for data efficiency, thereby incurring minimal additional computational overhead. We identify and formalize four key guidances for optimizing data organization: Boundary Sharpening, Cyclic Scheduling, Curriculum Continuity, and Local Diversity. Guided by them, we introduce two novel data ordering methods termed STR and SAW. Extensive experiments across different model scales and data sizes, encompassing both pre-training and SFT stages, validate the effectiveness of our summarized guidances. They also demonstrate the robustness of our proposed data ordering methods in enhancing the stability and performance of LLM training.


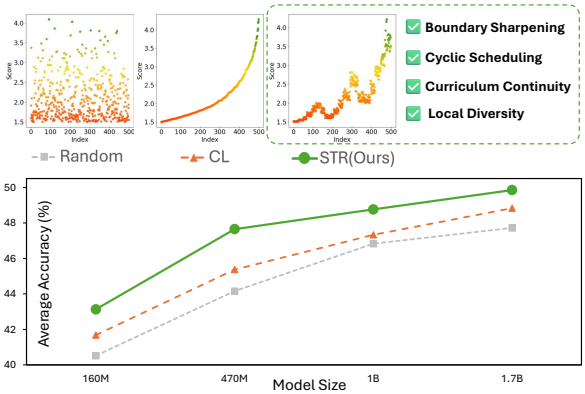


Figure 1: Performance comparison of different data organization strategies. Bottom: Average accuracy (%) under increasing model sizes (160M–1.7B). Our proposed SAW,consistently outperform Random and Curriculum Learning (CL) across all scales, with larger gains at higher capacities. Top: Score–index distributions of training data organized by different strategies. Compared to Random and CL, SAW induces more structured and progressive score patterns, revealing increasingly organized curricula.

## 1 Introduction

Large Language Models (LLMs) have revolutionized various fields by providing unprecedented capabilities in areas such as coding (Luo et al., 2023; Yu et al., 2024), science (Van Noorden and Perkel, 2023), nature (Huang et al., 2025), and healthcare (Väänänen et al., 2021). While data curation strategies, including acquisition (Barbaresi, 2021; Bevendorff et al., 2023; Chang et al., 2024), mixing (Albalak et al., 2023; Ge et al., 2025; Ye et al., 2024), synthesis (Chen et al., 2025; Zhou et al., 2024), deduplication (Abbas et al., 2023; Tirumala et al., 2023), filtering (Li et al., 2024b; Goyal et al., 2024; Thrush et al., 2024), and selection (Albalak et al., 2024; Xie et al., 2023; Dai et al., 2025b; Gu et al., 2025), have been extensively studied and significantly advanced LLM development.

However, a critical but underexplored aspect of LLM training efficacy lies *not merely in what data is used, but how it is presented in a given dataset.*

Despite the existence of well-curated datasets, the strategic organization of these training samples remains a significant challenge. Current LLMs are often trained for only one or a few epochs over massive corpora (OpenAI; Dubey et al., 2024; Team et al., 2023; Yang et al., 2025). Under such paradigms, the temporal order in which training samples are presented becomes a first-order factor shaping the learning trajectory. Existing approaches like Curriculum Learning (Bengio et al., 2009), which arranges data samples by difficulty, and DELT (Dai et al., 2025a), which employs a folding learning strategy to look back at the similar scoring distribution of training data and offer

valuable insights into data organization. Yet, none of these works provide a systematic exploration of comprehensive guidance specifically tailored for data ordering in LLM training.

To bridge this critical gap, this paper presents the first systematic exploration into the influence of data organization on LLM training. A key innovation of our approach is the reuse of pre-computed, sample-level scores, originally generated for data efficiency (Lozhkov et al., 2024; Wettig et al., 2024), which incurs almost zero additional computational overhead. Through extensive analysis, we identify and formalize four novel and critical guidances for optimizing data organization:

- **Boundary Sharpening.** Start training with low-score data to stabilize early learning, and finish with high-score data to enhance the model's performance. This approach helps the model handle complex reasoning tasks effectively.

- **Cyclic Scheduling.** Instead of a strict low-to-high score curriculum that can cause forgetting, cyclic scheduling periodically reintroduces a mix of data distribution. This helps the model retain fundamental knowledge and prevents forgetting.

- **Curriculum Continuity.** Smooth transitions between different score distributions of data prevent disruptions in training, maintaining stability and ensuring the model adapts progressively without losing momentum.

- **Local Diversity.** Mixing diverse samples within mini-batches prevents overfitting and promotes learning of generalizable features, improving the model's ability to generalize from the data.

To demonstrate how these guidances can be employed in practice, we instantiate them into two novel ordering strategies termed STR and SAW, which systematically integrate these guidances to create an optimized training sequence. The score-index distributions of these organized sequences are presented in the bottom part of Figure 1. Extensive experiments across diverse model scales and data sizes, encompassing both pre-training and supervised fine-tuning (SFT) stages, confirm the effectiveness of our formalized guidances. The top part of Figure 1 illustrates the efficacy of the proposed STR and SAW methods in improving LLM training stability and overall performance.

## 2 Problem Formulation

Training large language models (LLMs) with vast datasets from the web is costly. To improve data efficiency, previous methods spend a lot of resources calculating scores for each data sample (e.g., quality, difficulty, learnability, educational value), but use them just once for selecting data. In contrast, this work aims to reuse these scores to organize training data, improving model performance. We formalize this process in three stages:

**Data Scoring.** This stage evaluates the raw dataset $\mathcal{D} = \{x_1, \ldots, x_{|\mathcal{D}|}\}$ to derive fine-grained metrics. Let $g$ denote the scoring function that maps $\mathcal{D}$ to a score vector $\boldsymbol{\gamma} \in \mathbb{R}^{|\mathcal{D}|}$, representing specific criteria such as quality or difficulty:

$$\boldsymbol{\gamma} = g(\mathcal{D}) = [\gamma_1, \gamma_2, \ldots, \gamma_{|\mathcal{D}|}]^\top. \tag{1}$$

**Data Selection.** This process identifies an optimal subset to reduce training costs. Let $f_s$ be the selection function with a selection ratio $R$. It filters $\mathcal{D}$ based on $\boldsymbol{\gamma}$, retaining the top-$K$ samples where $K = \lfloor R \cdot |\mathcal{D}| \rfloor$. The resulting subset $\mathcal{D}_{\text{sub}}$ changes the dataset scale but does not inherently determine the training order:

$$\mathcal{D}_{\text{sub}} = f_s(\mathcal{D}; \boldsymbol{\gamma}, K) = \{x_i \in \mathcal{D} \mid r(\gamma_i) \leq K\}, \tag{2}$$

where $r(\cdot)$ indexes elements in descending order of their scores $\boldsymbol{\gamma}$.

**Data Organization.** This is the core part of our work, focusing on the strategic arrangement of data. Unlike selection, Data Organization $f_o$ does not alter the size but reorganizes the sequence of the dataset based on a permutation $\pi$ derived from the scores $\boldsymbol{\gamma}$. As a result, the initially disordered data is transformed into an ordered data:

$$\mathcal{D}_{\text{ord}} = f_o(\mathcal{D}; \boldsymbol{\gamma}) = [x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(K)}]. \tag{3}$$

where the permutation $\pi$ effectively translates the score into an ordering index for each data point. Finally, the overall pipeline is formalized as $\mathcal{D}_{\text{train}} = f_o(f_s(\mathcal{D}; \boldsymbol{\gamma}, K); \boldsymbol{\gamma})$, $f_o$ operates on either the full dataset or a selected subset. To maximize the utility of the pre-computed scores, our pipeline employs the *identical* score vector $\boldsymbol{\gamma}$ to guide both data selection and organization, ensuring the computational effort in scoring is used without additional overhead. As a special case of data organization, Curriculum Learning (CL) corresponds to an ordering function $f_o$ that sorts samples in ascending order of scores $\boldsymbol{\gamma}$, yielding $\mathcal{D}_{\text{sort}}$.

## 3 Guidances for Data Organization

### 3.1 G1: Boundary Sharpening.

**Motivation.** The training trajectory of LLMs is highly sensitive to data distribution at the optimization boundaries, particularly at the start and end of training. Recent findings indicate that improper data characteristics at initialization can destabilize early optimization and hinder convergence (Luo and Yang, 2023; Kalra and Barkeshli, 2024; Paul et al., 2022; Zhang et al., 2025). Likewise, the data distribution at the end of training greatly affects the model's attainable performance. Insufficient quality or complexity restricts performance on downstream tasks (Hu et al., 2024; Dubey et al., 2024). Thus, controlling data scoring distribution at these phases is crucial for optimal training dynamics.

**Guidance.** Boundary Sharpening advocates explicitly controlling the data characteristics at the start and end of training.

**Algorithm 1** Segment Ordering (SEG)

**Input:** Sorted dataset $\mathcal{D}_{\text{sort}}$, intervals $\{I_l\}_{l=0}^{L-1}$
**Output:** Organized dataset $\mathcal{D}_{\text{ord}}$
**Desc.:** Percentile-based partitioning & shuffling

```
{D_l}_{l=0}^{L-1} ← ∅
for all x in D_sort do
    k ~ Uniform({l | rank(x) ∈ I_l})
    D_k ← D_k ∪ {x}
end for
{D̃_l}_{l=0}^{L-1} ← Shuffle{D_l}_{l=0}^{L-1}
D_ord ← Concatenate(D̃_0, D̃_1, ..., D̃_{L-1})
```

**Implementation (Alg. 1).** To instantiate Boundary Sharpening in practice, we propose a **Segment Ordering (SEG)** strategy that discretizes the training sequence into $L$ distinct segments, $\mathcal{D}_0, \ldots, \mathcal{D}_{L-1}$, derived from the sorted data $\mathcal{D}_{\text{sort}}$. Each segment $\mathcal{D}_l$ is defined by a target percentile interval $[p_{\text{start}}^{(l)}, p_{\text{end}}^{(l)}]$ of $\mathcal{D}_{\text{sort}}$. To construct the curriculum, we map samples to segments based on their score rankings. To prevent duplication in overlapping regions, shared samples are evenly distributed among segments. Each segment $\mathcal{D}_l$ is then randomly shuffled to create $\tilde{\mathcal{D}}_l$. The final organized dataset is constructed by concatenating these processed segments: $\mathcal{D}_{\text{ord}} = [\tilde{\mathcal{D}}_0, \tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_{L-1}]$.

### 3.2 G2: Cyclic Scheduling.

**Motivation.** While a strict scoring-based sorting is intuitive, it poses risks in the one-pass training regime typical of LLMs (Tay et al., 2019; Tudor Ionescu et al., 2016). As the model transitions exclusively to complex samples in later stages, it tends to forget fundamental knowledge acquired from early simple data (Wang et al., 2021; Dai et al., 2025a). Standard solutions like Baby Step methods (Spitkovsky et al., 2010; Cirik et al., 2016), which mix previous simple data with new hard data, are often impractical in LLMs training as they require explicit data replay, inflating the training costs.

**Guidance.** Cyclic Scheduling advocates periodically revisiting training samples across the full score spectrum during one-pass training.

**Algorithm 2** Folding Ordering (FO)

**Input:** Sorted dataset $\mathcal{D}_{\text{sort}}$, folding layers $L$
**Output:** Organized dataset $\mathcal{D}_{\text{ord}}$
**Desc.:** Strided partitioning of sorted data

```
for l = 0 to L − 1 do
    D_l ← {D_sort[i] | 0 ≤ i ≤ |D_sort − 1|, i ≡ l (mod L)}
end for
D_ord ← Concatenate(D_0, D_1, ..., D_{L-1})
```

**Implementation (Alg. 2).** To practically instantiate Cyclic Scheduling, we introduce a **Folding Ordering (FO)** strategy based on pre-computed scores $\boldsymbol{\gamma}$. Let $\pi_{\text{sort}}$ be the permutation indices of sorted dataset $\mathcal{D}_{\text{sort}}$, such that $\gamma_{\pi_{\text{sort}}(0)} \leq \cdots \leq \gamma_{\pi_{\text{sort}}(|D|-1)}$. We divide the training dataset into $L$ folding layers. To ensure each folding cycle covers the entire score spectrum, we distribute the sorted samples in a strided partitioning way. Specifically, the dataset is reorganized into a series of cycles $\mathcal{D}_0, \ldots, \mathcal{D}_{L-1}$. Each cycle corresponds to a folding layer in the training sequence, and the $l$-th folding layer $\mathcal{D}_l$ $(0 \leq l < L)$ collects samples with sorted ranks congruent to $l$ modulo $L$:

$$\mathcal{D}_l = \left[x_{\pi_{\text{sort}}(i)} \mid i \equiv l \pmod{L},\ 0 \leq i \leq |\mathcal{D}| - 1\right]. \quad (4)$$

The final ordered dataset $\mathcal{D}_{\text{ord}}$ is created by concatenating these cycles: $\mathcal{D}_{\text{ord}} = [\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{L-1}]$. Within each cycle $\mathcal{D}_l$, samples keep their relative low-to-high score order. This construction effectively transforms a globally increasing trajectory into a periodic pattern, allowing the model to revisit early basic concepts regularly while progressively tackling samples with higher scores.

### 3.3 G3: Curriculum Continuity.

**Motivation.** The dynamics of the optimization process are highly sensitive to the sequence of data presentation. Abrupt fluctuations in data attributes may induce shocks to the optimizer, manifesting as rapid switching between relatively low and high-scoring samples(Kong et al., 2021; Weinshall et al., 2018). Such shifts often lead to high variance in gradient estimates, disrupting the optimization stability and potentially causing loss divergence.

**Guidance.** Curriculum Continuity advocates maintaining smooth transitions in data attributes throughout the training sequence.

**Algorithm 3** Zig-zag Ordering (ZIG)

**Input:** Sorted dataset $\mathcal{D}_{\text{sort}}$, folding layers $L$
**Output:** Organized dataset $\mathcal{D}_{\text{ord}}$
**Desc.:** Enhancing continuity across partitions
$\{\mathcal{D}_l\}_{l=0}^{L-1} \leftarrow \text{FO}(\mathcal{D}_{\text{sort}}, L)$ {See Alg. 2}
**for** $l = 0$ **to** $L-1$ **do**
  **if** $l$ is odd **then**
    $\mathcal{D}_l \leftarrow \text{Reverse}(\mathcal{D}_l)$
  **end if**
**end for**
$\mathcal{D}_{\text{ord}} \leftarrow \text{Concatenate}(\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{L-1})$

**Implementation. (Alg. 3)** To instantiate Curriculum Continuity in practice, we propose the **Zig-zag Ordering (ZIG)** strategy. FO (Sec. 3.2) causes sharp attribute drops when switching between cycles, violating the continuity required for stable optimization. ZIG resolves this by simply reversing the sample order in every odd-indexed cycle (i.e., $\mathcal{D}_1, \mathcal{D}_3, \ldots$) from Eq. 4. The final dataset $\mathcal{D}_{\text{ord}}$ concatenates these partially reversed subsets to a continuous triangle-wave pattern, ensuring that adjacent cycles typically connect at points of similar score and yielding a smoother training trajectory.

### 3.4 G4: Local Diversity.

**Motivation.** While establishing a global curriculum (e.g., via cyclic scheduling or curriculum continuity) is beneficial, strictly sorting the dataset by scores $\gamma$ negatively impacts *Local Diversity*. When training samples within a mini-batch possess nearly identical scores and attributes, the diversity of gradients diminishes (Jiang et al., 2014). This lack of intra-batch variance may bias the optimizer towards specific, repetitive patterns, leading to overfitting and reducing the model's ability to learn generalizable features (Yin et al., 2018).

**Guidance.** Local Diversity advocates preserving sufficient heterogeneity among training samples within local sections of the training sequence.

**Algorithm 4** Jittering Ordering (JIT)

**Input:** Sorted dataset $\mathcal{D}_{\text{sort}}$, window size $w$
**Output:** Organized dataset $\mathcal{D}_{\text{ord}}$
**Desc.:** Injecting index noise for local diversity
$L \leftarrow \lceil |\mathcal{D}_{\text{sort}}|/w \rceil$
**for** $l = 0$ **to** $L-1$ **do**
  $\mathcal{D}_l \leftarrow \text{Shuffle}(\mathcal{D}_{\text{sort}}[l \cdot w : (l+1) \cdot w])$
**end for**
$\mathcal{D}_{\text{ord}} \leftarrow \text{Concatenate}(\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{L-1})$

**Implementation. (Alg. 4)** To instantiate Local Diversity in practice, we introduce **Jittering Ordering (JIT)** strategy by injecting tunable noise into the strictly sorted data $\mathcal{D}_{\text{sort}}$ (e.g., CL, FO, and ZIG). Specifically, we partition the sorted dataset $\mathcal{D}_{\text{sort}}$ into contiguous buckets of size $w$ (diversity window). Within each bucket, we apply random shuffling while preserving the relative order between buckets. The final dataset $\mathcal{D}_{\text{ord}}$ concatenates these locally shuffled buckets, restoring essential gradient noise while maintaining the global trend.

## 4 Methodology

Besides discussing each guidance individually, we also propose several cross-guidance strategies to explore their interplay (Alg. 5).

**Cross-guidance strategies.** A cross-guidance strategy combines multiple guidances into one data ordering function by assigning different ordering mechanisms to different parts of the training sequence. This approach enables controlled composition of global progression, periodic review, continuity, and local diversity within a unified framework, rather than relying on a single ordering heuristic.

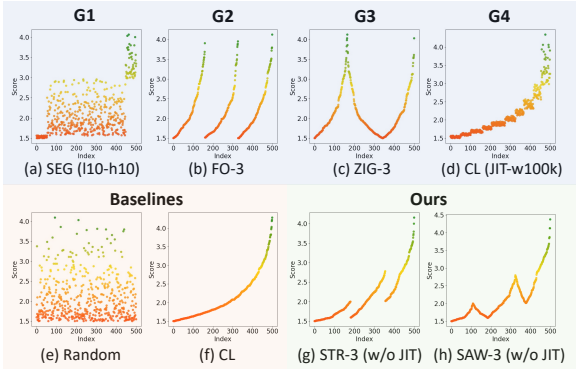


Figure 2: Visualization of score-index distribution under different data organization strategies. For clear comparison, $L = 3$ is used for FO, ZIG, STR, and SAW.

**Algorithm 5** STR & SAW

**Input:** Sorted dataset $\mathcal{D}_{\text{sort}}$, split points $\{p_l\}_{l=1}^{L-1}$, radius $\rho$, window size $w$, folding layers $L$, mode $T \in \{\text{STR}, \text{SAW}\}$
**Output:** Organized dataset $\mathcal{D}_{\text{ord}}$
**Desc.:** Cross-guidance strategy.
$\{\mathcal{D}_l^t\}_{l=1}^{L-1} \leftarrow \{\mathcal{D}_{\text{sort}}[p_l - \rho : p_l + \rho]\}$
$\{\mathcal{D}_l^s\}_{l=0}^{L-1} \leftarrow \{\mathcal{D}_{\text{sort}}[p_l + \rho : p_{l+1} - \rho]\}$
with $p_0 = -\rho, p_L = |\mathcal{D}_{\text{sort}}| + \rho$
**for** each $\mathcal{D}_l^t \in \{\mathcal{D}_l^t\}$ **do**
  **if** $T = \text{STR}$ **then**
    $\mathcal{D}_l^t \leftarrow \text{FO}(\mathcal{D}_l^t, L)$ {G2, see Alg. 2}
  **else if** $T = \text{SAW}$ **then**
    $\mathcal{D}_l^t \leftarrow \text{ZIG}(\mathcal{D}_l^t, L)$ {G3, see Alg. 3}
  **end if**
**end for**
$\mathcal{D}_{\text{ord}} \leftarrow \text{Concatenate}(\mathcal{D}_0^t, \mathcal{D}_0^s, \ldots, \mathcal{D}_{L-1}^s, \mathcal{D}_{L-1}^t)$
**if** $w > 0$ **then**
  $\mathcal{D}_{\text{ord}} \leftarrow \text{JIT}(\mathcal{D}_{\text{ord}}, w)$ {G4, see Alg. 4}
**end if**

**Stair Ordering (STR).** This strategy follows guidances of **G1, G2, and G4**. To explore the benefits of global progression (G1) and periodic review (G2), STR maintains a global trend while injecting controllable folding mechanisms at local transitions to facilitate early knowledge review. Building upon this, we optionally apply the JIT mechanism to data samples to enhance local diversity (G4).

Specifically, we partition the $\mathcal{D}_{\text{sort}}$ into $L$ sections by identifying $L-1$ split points. Around each point, we define a transition region $\mathcal{D}^t$ with a radius $\rho$, while the remaining portions constitute stable regions $\mathcal{D}^s$. In the final sequence, stable regions retain their monotonic order, whereas transition regions apply the interleaving logic $f_{\text{FO}}$ (Eq. 4) to facilitate knowledge review.

**Saw Ordering (SAW).** SAW generalizes STR by integrating **G1, G2, G3, and G4**, and explicitly enforces continuity constraints during region transitions. While the STR ordering strategy effectively integrates G1&2, the use of folding mechanisms to transition between regions introduces abrupt shifts in data attributes, posing a challenge to G3 described in 3.3. To address this and explore the benefits of attribute continuity, SAW refines the STR framework by replacing the folding $f_{\text{FO}}$ within transition regions with the Zig-zag mechanism $f_{\text{ZIG}}$, thereby ensuring a smoother transition between samples at region boundaries. Similarly, we also optionally apply the JIT mechanism (G4).

# 5 Experiment

## 5.1 Experimental Setup

**Data.** For general data, we report FineWeb-Edu (Lozhkov et al., 2024) results in the main text and defer QuRatedPajama (Wettig et al., 2024) to Appendix E. For specific tasks, we utilize DeepMath-103K (He et al., 2025) for math reasoning and OpenCodeInstruct (Ahmad et al., 2025) for code generation. See Appendix C.2 for dataset details.

**Models.** We apply the **Mistral** (Jiang et al., 2023) architecture for pre-training on general data, and the **Qwen3** (Yang et al., 2025) for SFT in the math and code data using the official pre-trained weights. See Appendix C.3 for model details.

**Baselines.** We use random sampling (**Random**) and vanilla curriculum learning (**CL**) as baselines. Following Sec. 3, we verify the effectiveness of each guidance through specific implementations: **SEG** (G1), **FO** (G2), **ZIG** (G3), and **JIT** (G4). Besides, we investigate the cross-guidance strategies of the proposed **STR** and **SAW** in Sec. 4. These strategies are visualized in Figure 2. See Appendix C.4 for training details.

For additional details of the experimental setup, such as training and evaluation, see Appendix C.

## 5.2 Guidance Analysis

Refer to Appendix C.1 for setup of this section.

### 5.2.1 G1: Boundary Sharpening

**Results.** For the pre-training setting, as demonstrated in Table 1, we derive two observations: (1) *High-scoring data at the end optimizes performance.* Regardless of the initial phase, configurations ending with high-scoring samples (e.g., SEG(l10-h10)) consistently yield significant gains. Conversely, concluding with low-scoring data leads to marked performance degradation (e.g., SEG(h90)). (2) *Initialization with high-score data alone yields negligible benefits.* Configurations that exclusively prioritize high scores at the start (e.g., SEG(h10)) perform comparably to the random baseline. This may be attributed to the constraints of a fixed data volume, where selecting high-scoring samples early on inevitably leaves lower-quality data for the final stages.

For the SFT setting, the best performance is achieved when both the beginning and the end of the training utilize high-scoring data (SEG(h10-h10)). This stems from the fact that starting with high-scoring data stabilizes the transition from pre-training, while ending with high-scoring data consistently optimizes final performance.

**Analysis.** Figure 3 illustrates how data attributes at training boundaries shapes model dynamic performance. Variants ending with high-score data (e.g., SEG(l90), SEG(l10-h10)) exhibit a sharp final performance boost, surpassing the baseline. In contrast, those ending with low-score data (e.g., SEG(h90), SEG(h10-l10)) suffer from stagnation.

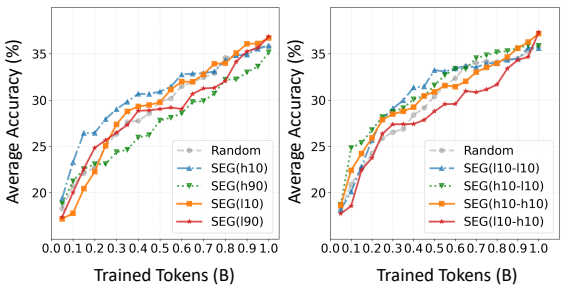


Figure 3: Comparison of accuracy trajectories for SEG. Results on Mistral-160M trained on 1B-tokens data.

Table 1: Performance comparison of SEG variants and baselines to evaluate G1.

| | **Pre-training** (FineWeb-Edu) | | | | | | | | | **SFT** (DeepMath-103K \| OpenCodeInstruct) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | AIME24 | AIME25 | Avg. | HumanEval | MBPP | Avg. |
| Random | 22.53 ± 0.18 | 35.44 ± 0.23 | 27.87 ± 0.15 | 14.63 ± 0.12 | 24.40 ± 0.20 | 56.37 ± 0.51 | 53.50 ± 0.48 | 51.54 ± 0.04 | 35.78 ± 0.08 | 1.29 ± 0.02 | 1.30 ± 0.02 | 1.30 ± 0.01 | 57.93 ± 0.60 | 52.80 ± 0.06 | 55.37 ± 0.30 |
| SEG(h10) | 22.01 | 34.09 | 27.72 | **13.51** | 27.40 | 57.34 | 55.20 | 49.80 | 35.89 ↑ | 1.89 | **2.55** | 2.22 | **66.40** | 52.00 | 59.20 |
| SEG(h90) | 21.08 | 33.80 | 27.38 | 12.48 | 25.00 | 56.96 | 52.40 | **52.25** | 35.17 ↓ | 2.10 | 1.82 | 1.96 | 59.10 | 53.20 | 56.15 |
| SEG(l10) | 22.01 | 36.57 | **28.05** | 13.12 | 26.60 | **58.11** | 57.50 | 51.78 | 36.72 ↑ | 2.53 | 1.15 | 1.84 | 64.60 | **54.20** | **59.40** |
| SEG(l90) | **24.15** | **37.79** | 27.90 | 12.79 | **27.80** | 56.04 | **57.90** | 50.43 | **36.85** ↑ | **2.64** | 2.29 | **2.47** | 59.10 | 53.00 | 56.05 |
| SEG(h10-l10) | 21.59 | 34.01 | 27.80 | **14.03** | 25.60 | **57.73** | 55.80 | 50.43 | 35.87 ↑ | 2.32 | 2.03 | 2.18 | 60.30 | 53.40 | 56.85 |
| SEG(l10-l10) | 22.01 | 33.92 | 27.39 | 13.16 | 23.40 | 57.29 | 55.10 | **52.57** | 35.60 ↓ | 2.05 | 1.67 | 1.86 | 63.40 | 52.60 | 58.00 |
| SEG(l10-h10) | **23.81** | **38.30** | **28.22** | 12.56 | 27.40 | 57.40 | **60.20** | 50.67 | **37.32** ↑ | 1.89 | 1.61 | 1.75 | 59.10 | 53.40 | 56.25 |
| SEG(h10-h10) | 22.53 | 37.58 | 27.99 | 13.58 | **28.60** | 57.13 | 59.40 | 50.59 | 37.18 ↑ | **2.37** | **2.14** | **2.26** | **66.40** | **54.60** | **60.50** |

Table 2: Performance comparison of FO variants and baselines to evaluate G2.

| | **Pre-training** (FineWeb-Edu) | | | | | | | | | **SFT** (DeepMath-103K \| OpenCodeInstruct) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | AIME24 | AIME25 | Avg. | HumanEval | MBPP | Avg. |
| Random | 22.53 ± 0.18 | 35.44 ± 0.23 | 27.87 ± 0.15 | **14.63** ± 0.12 | 24.40 ± 0.20 | 56.37 ± 0.51 | 53.50 ± 0.48 | 51.54 ± 0.04 | 35.78 ± 0.08 | 1.29 ± 0.02 | 1.30 ± 0.02 | 1.30 ± 0.01 | 57.93 ± 0.60 | 52.80 ± 0.06 | 55.37 ± 0.30 |
| CL | 22.87 | **39.56** | 28.05 | 10.89 | **29.40** | 55.66 | 58.20 | 50.83 | 36.93 | 1.94 | 1.61 | 1.78 | 60.90 | 51.40 | 56.15 |
| FO-2 | 24.15 | 38.38 | 28.09 | 10.69 | **29.40** | 55.82 | 61.20 | 51.07 | **37.35** ↑ | 1.89 | 1.56 | 1.73 | 60.90 | 53.60 | 57.25 |
| FO-3 | 23.46 | 38.64 | 27.71 | 10.58 | 27.60 | 56.47 | **61.70** | 50.36 | 37.07 ↑ | **2.96** | **1.88** | **2.42** | 65.80 | 53.40 | 59.60 |
| FO-4 | **24.76** | 37.26 | 28.00 | 11.01 | 26.90 | **57.77** | 58.20 | 51.88 | 36.97 ↑ | 2.16 | 1.72 | 1.94 | 62.10 | **55.60** | 58.85 |
| FO-5 | 23.29 | 38.68 | **28.16** | 11.86 | 26.80 | 56.47 | 59.60 | **52.17** | 37.13 ↑ | 2.80 | 1.20 | 2.00 | **66.80** | 54.80 | **60.80** |
| FO-20 | 23.46 | 36.99 | 27.66 | 13.62 | 26.60 | 56.37 | 56.80 | 51.22 | 36.59 ↓ | 2.42 | 1.61 | 2.02 | 60.90 | 51.20 | 56.05 |
| FO-100 | 22.53 | 35.86 | 27.57 | 11.43 | 24.20 | 56.91 | 53.20 | 47.75 | 34.93 ↓ | 2.48 | 1.51 | 2.00 | 66.40 | 51.80 | 59.10 |

Table 3: Performance comparison of ZIG variants and baselines to evaluate G3.

| | **Pre-training** (FineWeb-Edu) | | | | | | | | | **SFT** (DeepMath-103K \| OpenCodeInstruct) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | AIME24 | AIME25 | Avg. | HumanEval | MBPP | Avg. |
| Random | 22.53 ± 0.18 | 35.44 ± 0.23 | 27.87 ± 0.15 | 14.63 ± 0.12 | 24.40 ± 0.20 | 56.37 ± 0.51 | 53.50 ± 0.48 | 51.54 ± 0.04 | 35.78 ± 0.08 | 1.29 ± 0.02 | 1.30 ± 0.02 | 1.30 ± 0.01 | 57.93 ± 0.60 | 52.80 ± 0.06 | 55.37 ± 0.30 |
| FO-2(or 5) | 24.15 | 38.38 | 28.09 | 10.69 | **29.40** | 55.82 | **61.20** | 51.07 | 37.35 | 2.80 | 1.72 | 2.26 | 63.10 | **55.60** | 59.35 |
| ZIG-2(or 5) | **25.17** | **38.59** | **28.12** | **11.53** | 27.80 | **56.75** | 60.90 | **52.01** | **37.61** ↑ | **2.88** | **2.02** | **2.45** | **63.80** | 55.40 | **59.60** |
| FO-3 | **23.46** | 38.64 | 27.71 | 10.58 | **27.60** | 56.47 | **61.70** | 50.36 | 37.07 | **2.82** | 1.88 | 2.35 | **65.80** | 52.40 | 59.10 |
| ZIG-3 | 23.38 | **39.69** | **27.77** | **11.08** | 27.40 | **57.40** | 61.30 | **51.14** | **37.40** ↑ | 2.64 | **2.20** | **2.42** | 64.63 | **54.20** | **59.42** |

### 5.2.2 G2: Cyclic Scheduling

**Results.** For the pre-training stage, as shown in Table 2, both CL and FO methods achieve varying degrees of improvement over the random baseline. The best variant (FO-2) consistently exceeds the CL baseline, proving that the periodic review of early foundational knowledge offers benefits to the learning process. For the SFT stage, FO yields substantial gains, especially with FO-3 in mathematics and FO-4/5 in coding. This confirms that review strategies remain effective in SFT stage. However, the efficacy of FO across both stages is sensitive to the parameter $L$, as suboptimal selections can lead to performance degradation.

**Analysis.** Figure 4 presents the training-wise Perplexity (PPL) trends measured on the data with the 10% lowest score, referred to as $D_e$, within $D_{\text{sort}}$. The PPL of CL on $D_e$ rapidly drops (initial 30% phase) but rebounds after entering the high-score data region (latter 50%), which directly confirms the forgetting of early samples. For the FO-3, PPL drops normally in cycle 1. When simple data is reintroduced in cycle 2, PPL shows a secondary sharp drop. By the end of training (cycle 3), the model maintains a low PPL on $D_e$ and exhibits no rebound phenomenon seen in CL.

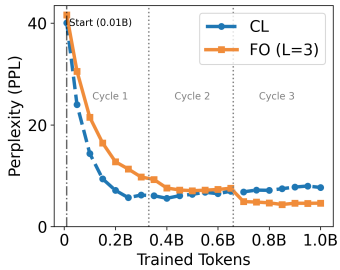


Figure 4: The LMs' perplexity (PPL) for $D_e$. Results on Mistral-160M trained on 1B-tokens data.

### 5.2.3 G3: Curriculum Continuity

**Results.** In Table 3, for the pre-training stage, the ZIG variants consistently outperform their FO counterparts. For the SFT stage, ZIG remains comparable to or outperforms FO in most cases. These results indicate that abrupt shifts in data attributes can negatively impact training, while enhancing attribute continuity effectively mitigates such issues.

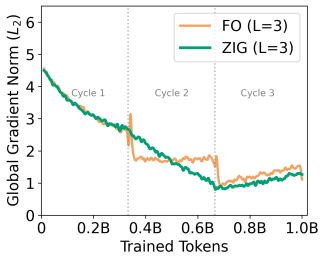


Figure 5: Comparison of gradient norm trajectories. Results on Mistral-160M trained on 1B-tokens data.

**Analysis.** To empirically investigate the impact of abrupt data attribute shifts on optimization stability, we visualize the global gradient norm. As shown in Figure 5, FO-3 exhibits a sharp spike in gradient norm at the boundary between cycles, indicating a shock to the optimizer due to the abrupt transition from samples with high to low score. In contrast, ZIG maintains a stable gradient magnitude, verifying its ability to to ensure smoother optimization dynamics and stabilize the training process.

#### 5.2.4 G4: Local Diversity

**Results.** For both pre-training and SFT stages, as indicated in Table 4, the application of the JIT strategy leads to consistent performance gains across all strictly sorted ordering methods. This observation suggests that the lack of local diversity acts as an inherent limitation in model training, which can be effectively mitigated by the JIT strategy to further improve model performance.

Table 4: Performance comparison of JIT variants and baselines to evaluate G4.

| | FineWeb-Edu | QuRatedPajama | DeepMath | OpenCodeInstruct |
|---|---|---|---|---|
| Random | 35.78 ± 0.08 | 35.60 ± 0.16 | 1.30 ± 0.02 | 55.47 ± 0.22 |
| CL | 36.93 | 36.12 | 1.78 | 58.30 |
| CL (JIT) | **37.72** ± 0.14 | **36.46** ± 0.11 | **1.78** ± 0.03 | **59.50** ± 0.35 |
| FO | 37.35 | 36.62 | 2.42 | 60.90 |
| FO (JIT) | **37.42** ± 0.09 | **36.85** ± 0.09 | **2.74** ± 0.12 | **60.96** ± 0.14 |
| ZIG | 37.61 | 36.74 | 2.69 | 60.11 |
| ZIG (JIT) | **37.65** ± 0.12 | **36.88** ± 0.10 | **2.76** ± 0.08 | **61.34** ± 0.18 |

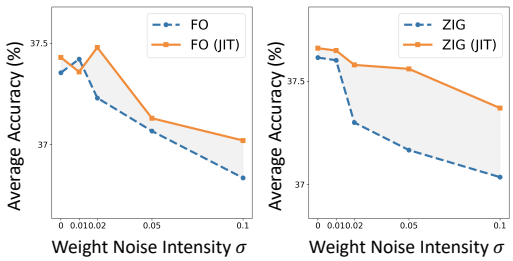


Figure 6: Sensitivity analysis of model performance to weight perturbation.

**Analysis.** Previous works have shown that the flatness of the loss landscape is strongly correlated with generalization and robustness (Keskar et al., 2016; Foret et al., 2020). To evaluate how JIT enhances model robustness, we conduct weight perturbation analysis by injecting multiplicative Gaussian noise into parameters of various models (e.g., CL, FO, ZIG) and their JIT-enhanced version.

As illustrated in Figure 6, the results show that (1) *Superior performance.* Models with JIT consistently outperform baselines across all noise levels. (2) *Flatter minima.* While the performance of baselines drops quickly as the noise scale $\alpha$ increases, JIT-enhanced models exhibit slower degradation. This lower sensitivity shows that JIT helps the model find broader and flatter minima, leading to better robustness and generalization potential.

### 5.3 Main Result

**Setup.** To investigate the interplay among different guidances, we design the STR (Guidance 1&2&4) and SAW (Guidance 1&2&3&4) strategies (see Sec. 4). Data organization methods of CL (Bengio et al., 2009) and DELT (Dai et al., 2025a) are employed as baselines. In the main text, we report the best performance across $L$ and JIT configurations.

**Results.** For both pre-training and SFT stages, as exhibited in Table 5, the cross-guidance strategies of STR and SAW significantly outperform the CL and DELT baselines. Moreover, SAW and STR exhibit comparable performance. This parity likely stems from the fact that, unlike the full restart review in FO, STR narrows the scope of the review data, naturally mitigating the drastic attribute shifts. Therefore, the additional continuity benefits (Guidance 3) introduced by SAW become less apparent.

### 5.4 Scaling-up Result

**Setup.** To assess the scalability and robustness of our ordering strategies, we evaluate our methods on a 50B-token data corpus across various Mistral model sizes, including 160M, 470M, 1B, and 1.7B parameters. We apply STR and SAW using the optimal configurations derived from the 160M/1B-token setup.

**Results.** As shown in Table 6, both STR and SAW exhibit strong performance as model and data scales increase, consistently outperforming the random baseline across downstream tasks on average. Beyond downstream tasks, we show that data organization also improves language modeling performance on high-quality corpora as the model

Table 5: Performance comparison of cross-guidance strategies (STR and SAW) against other strategies. 24 and 25 refer to the AIME24 and AIME25, while HE refers to the HumanEval benchmark.

| | Guidances | | | | Pre-training (FineWeb-Edu) | | | | | | | | | SFT (DeepMath-103K \| OpenCodeInstruct) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | G1 | G2 | G3 | G4 | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | 24 | 25 | Avg. | HE | MBPP | Avg. |
| Random | - | ✓ | - | ✓ | 22.53 | 35.44 | 27.87 | **14.63** | 24.40 | 56.37 | 53.50 | 51.54 | 35.78 | 1.29 | 1.30 | 1.30 | 57.93 | 53.00 | 55.47 |
| CL (Bengio et al., 2009) | ✓ | - | ✓ | - | 22.87 | 39.56 | 28.05 | 10.89 | 29.40 | 55.66 | 58.20 | 50.83 | 36.93 | 1.94 | 1.61 | 1.78 | 61.59 | 55.00 | 58.30 |
| DELT (Dai et al., 2025a) | ✓ | ✓ | - | - | 24.15 | 38.38 | 28.09 | 10.69 | 29.40 | 55.82 | 61.20 | 51.07 | 37.35 | 2.96 | 1.88 | 2.42 | 63.80 | 55.60 | 59.70 |
| STR (Ours) | ✓ | ✓ | - | ✓ | **25.59** | **40.40** | 27.93 | 10.85 | **29.70** | **56.61** | 60.80 | 49.65 | 37.69 | **3.02** | 1.93 | 2.48 | **65.85** | 55.80 | **60.83** |
| SAW (Ours) | ✓ | ✓ | ✓ | ✓ | 23.63 | 40.11 | **28.25** | 12.05 | 28.20 | 55.55 | **61.80** | **53.04** | **37.83** | 2.91 | **2.14** | **2.53** | 64.76 | **56.20** | 60.48 |

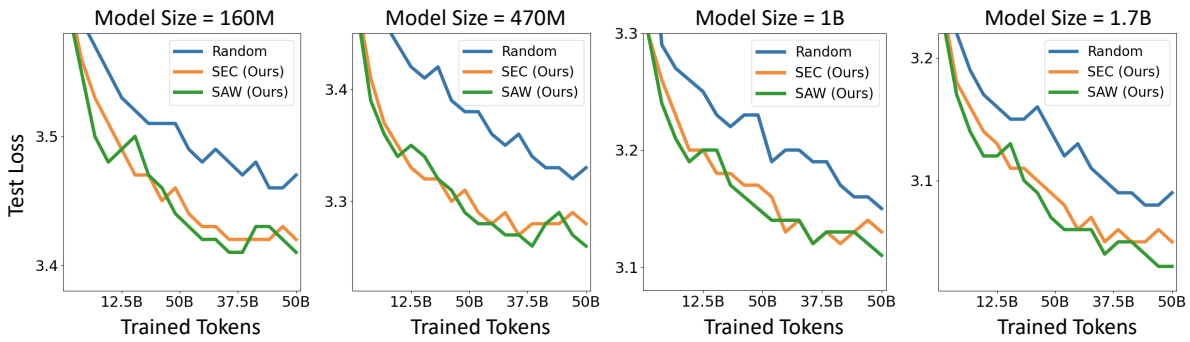


Figure 7: Test losses on the DCLM corpus (Li et al., 2024a) across 160M to 1.7B model sizes. The labels STR and SAW refer to the optimal configurations: STR-2(JIT) and SAW-2(JIT).

Table 6: Scaling-up of different cross-guidance strategies during the pre-training stage on two 50B-token training datasets. QR refers to the QuRatedPajama dataset, while FW refers the FineWeb-Edu dataset.

| | 160M | | 470M | | 1B | | 1.7B | |
|---|---|---|---|---|---|---|---|---|
| | QR | FW | QR | FW | QR | FW | QR | FW |
| Random | 39.83 | 40.52 | 43.50 | 44.16 | 45.96 | 46.83 | 47.28 | 47.72 |
| SEC | 41.27 | **43.13** | 44.75 | **47.65** | 46.33 | **48.45** | **48.76** | 49.85 |
| SAW | **41.51** | 43.10 | **45.07** | 46.83 | **46.59** | 48.06 | 48.42 | **50.11** |

Table 7: Test loss extrapolation via Chinchilla Scaling Law (Hoffmann et al., 2022). We estimate losses where model size $N$ and trained tokens $D$ align with GPT-3 175B, Llama 6.7B, Llama 2 70B, and Llama 3.1 405B. The improvements of data organization remain consistent for these LMs.

| | $N$ | $D$ | Random | STR | SAW |
|---|---|---|---|---|---|
| GPT-3 | 175B | 300B | 2.876 | **2.859** | **2.853** |
| Llama | 6.7B | 1.0T | 2.895 | **2.831** | **2.824** |
| Llama 2 | 70B | 2.0T | 2.788 | **2.759** | **2.753** |
| Llama 3.1 | 405B | 15T | 2.762 | **2.744** | **2.738** |

scale increases. Figure 7 illustrates the test loss on the DCLM subset, comparing conventionally pre-trained LMs (random) against those trained on data ordered by STR and SAW. The results show that LMs trained on ordered data constantly achieve better performance across various model sizes.

Table 7 details our extrapolation of test losses using the Chinchilla Scaling Law (Hoffmann et al., 2022), showing that the gains from ordered data persist even when pre-training recent large LMs such as GPT-3 (Brown et al., 2020) and the Llama family (Touvron et al., 2023a,b; Dubey et al., 2024). More details of the extrapolation are available in the Appendix D. The DCLM corpus is assembled using a complex heuristic-based pipeline and is confirmed to be both diverse and exhaustive in its knowledge representation. Utilizing existing scores from data selection for data organization comes at a negligible cost and introduces a new perspective for enhancing model performance compared to complex curation pipelines.

## 6 Conclusion

This paper systematically explores the impact of data organization on Large Language Model (LLM) training, an area often overlooked despite its significance, especially given the limited number of training epochs for current LLMs. By utilizing pre-computed sample-level scores from data efficiency methods, our approach introduces minimal computational overhead. We identified and formalized four key guidances for optimizing training data organization. According to them, two data ordering methods, STR and SAW, were proposed. Extensive experiments consistently validated the effectiveness of these data ordering guidances and strategies enhance training stability and model performance across various model scales and data sizes.

## Limitations

While this work presents a systematic investigation into data organization for LLM training and introduces effective guidances, it is important to acknowledge certain limitations. The methodology relies on reusing pre-computed sample-level scores, which, while offering the significant advantage of almost zero additional computational overhead, also means that the efficacy of the proposed organization strategies is contingent upon the quality and relevance of these pre-existing scores.

## Ethics Statement

We ensure transparency by utilizing publicly available data sources and model architectures. We adhere to the licensing terms for both models and datasets, including proper attribution and sharing any modifications or derivative works under compatible terms. While this work focuses on language modalities, further research is necessary for unbiased evaluation in other modalities.

## References


Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. 2025. Opencodeinstruct: A large-scale instruction tuning dataset for code llms. *arXiv preprint arXiv:2504.04030*.

AIME. 2025. AIME problems and solutions. Accessed: 2026-01-04.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, and 1 others. 2024. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. 2023. Efficient online data mixing for language model pre-training. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Adrien Barbaresi. 2021. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Janek Bevendorff, Sanket Gupta, Johannes Kiesel, and Benno Stein. 2023. An empirical comparison of web content extraction algorithms. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2594–2603, New York, NY, USA. Association for Computing Machinery.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of AAAI*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and 1 others. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS*.

Ernie Chang, Hui-Syuan Yeh, and Vera Demberg. 2021. Does the order of training samples matter? improving neural data-to-text generation with curriculum learning. *arXiv preprint arXiv:2102.03554*.

Yaoyao Chang, Lei Cui, Li Dong, Shaohan Huang, Yangyu Huang, Yupan Huang, Scarlett Li, Tengchao Lv, Shuming Ma, Qinzheng Sun, and 1 others. 2024. Redstone: Curating general, code, math, and qa data for large language models. *arXiv preprint arXiv:2412.03398*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Zui Chen, Tianqiao Liu, Mi Tian, Weiqi Luo, Zitao Liu, and 1 others. 2025. Advancing mathematical reasoning in language models: The impact of problem-solving data, data synthesis methods, and training stages. In *The Thirteenth International Conference on Learning Representations*.

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. 2016. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina

Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Yalun Dai, Yangyu Huang, Xin Zhang, Wenshan Wu, Chong Li, Wenhui Lu, Shijie Cao, Li Dong, and Scarlett Li. 2025a. Data efficacy for language model training. *arXiv preprint arXiv:2506.21545*.

Yalun Dai, Lingao Xiao, Ivor W Tsang, and Yang He. 2025b. Training-free dataset pruning for instance segmentation. *arXiv preprint arXiv:2503.00828*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. 2025. Bimix: A bivariate data mixing law for language model pretraining. *Preprint*, arXiv:2405.14908.

Sachin Goyal, Pratyush Maini, Zachary C Lipton, Aditi Raghunathan, and J Zico Kolter. 2024. Scaling laws for data filtering–data curation cannot be compute agnostic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22702–22711.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. OLMo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. 2025. Data selection via optimal control for language models. In *The Thirteenth International Conference on Learning Representations*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, and 1 others. 2022. Training compute-optimal large language models. In *Proceedings of NeurIPS*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Yangyu Huang, Tianyi Gao, Haoran Xu, Qihao Zhao, Yang Song, Zhipeng Gui, Tengchao Lv, Hao Chen, Lei Cui, Scarlett Li, and 1 others. 2025. Peace: Empowering geologic map holistic understanding with mllms. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3899–3908.

Peter J. Huber. 1992. *Robust Estimation of a Location Parameter*. Springer New York.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander G Hauptmann. 2014. Self-paced learning with diversity. *Advances in neural information processing systems*, 27.

Dayal Singh Kalra and Maissam Barkeshli. 2024. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37:111760–111801.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Jisu Kim and Juhwan Lee. 2024. Strategic data ordering: Enhancing large language model performance through curriculum learning. *arXiv preprint arXiv:2405.07490*.

Yajing Kong, Liu Liu, Jun Wang, and Dacheng Tao. 2021. Adaptive curriculum learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5067–5076.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of KR*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, and 1 others. 2024a. DataComp-LM: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024b. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Yaoru Luo and Ge Yang. 2023. Enhancing robustness of deep networks against noisy labels based on a two-phase formulation of their learning behavior. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1763–1768.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of EMNLP*.

OpenAI. hello-gpt-4o. (2024).

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of ACL*.

Mansheej Paul, Brett Larsen, Surya Ganguli, Jonathan Frankle, and Gintare Karolina Dziugaite. 2022. Lottery tickets on a data diet: Finding initializations with sparse trainable networks. *Advances in Neural Information Processing Systems*, 35:18916–18928.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only. *Advances in Neural Information Processing Systems*, 36:79155–79172.

Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759.

Yi Tay, Shuohang Wang, Luu Anh Tuan, Jie Fu, Minh C Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. 2024. Improving pretraining data using perplexity correlations. *arXiv preprint arXiv:2409.05816*.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2023. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36:53983–53995.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim P Papadopoulos, and Vittorio Ferrari. 2016. How hard can it be? estimating the difficulty of visual search in an image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2157–2166.

Antti Väänänen, Keijo Haataja, Katri Vehviläinen-Julkunen, and Pekka Toivanen. 2021. Ai in healthcare: A narrative review. *F1000Research*, 10:6.

Richard Van Noorden and Jeffrey M Perkel. 2023. Ai and science: what 1,600 researchers think. *Nature*, 621(7980):672–675.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020. Curriculum pre-training for end-to-end speech translation. *arXiv preprint arXiv:2004.10093*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International conference on machine learning*, pages 5238–5246. PMLR.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (ACL 2017)*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the twelfth language resources and evaluation conference*, pages 4003–4012.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. 2023. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36:34201–34227.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. 2024. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. 2018. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007. PMLR.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, Yishujie Zhao, Wenxiang Hu, and Qiufeng Yin. 2024. Wavecoder: Widespread and versatile enhancement for code large language models by instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5140–5153.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of ACL*.

Yang Zhang, Amr Mohamed, Hadi Abdine, Guokan Shang, and Michalis Vazirgiannis. 2025. Beyond random sampling: Efficient language model pretraining via curriculum learning. *arXiv preprint arXiv:2506.11300*.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. 2024. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. *arXiv preprint arXiv:2405.14365*.

# Appendix

## A Extended Discussion on Guidances

We offer additional insights and supplementary details regarding the proposed guidances to further clarify their mechanisms.

- **Boundary Sharpening.** Improper data characteristics at the start and end of training can hinder convergence (Luo and Yang, 2023; Kalra and Barkeshli, 2024; Paul et al., 2022) or restrict peak performance (Hu et al., 2024; Dubey et al., 2024). We establish a boundary sharpening guidance, initiating training with low-score (simple, low-information) data to stabilize early optimization, while concluding with high-score (complex, high-quality) samples to align the model's capabilities with downstream reasoning tasks, thereby ensuring robust and performant models.

- **Cyclic Scheduling.** In a one-pass training regime (Tay et al., 2019; Tudor Ionescu et al., 2016; Wang et al., 2020), a strict easy-to-hard data curriculum often leads to the forgetting of early fundamental knowledge as the model transitions to complex samples (Wang et al., 2021). Moreover, the common practice of training LLMs for only one or a few epochs makes Baby Step methods (Bengio et al., 2009; Spitkovsky et al., 2010; Cirik et al., 2016) impractical, given their requirement for replaying simple data whenever more challenging samples are introduced. To address this, we propose cyclic scheduling guidance, which periodically feeds the model with data covering a broad score distribution.

- **Curriculum Continuity.** Abrupt fluctuations in data scores can shock the optimizer, causing loss divergence or training instability (Kong et al., 2021; Weinshall et al., 2018). We effectively maintain curriculum continuity, ensuring smooth transitions across the spectrum of sample attributes to allow the model to progressively adapt to evolving data patterns without disrupting the optimization momentum, which is crucial for maintaining stable learning trajectories.

- **Local Diversity.** Strictly sorting data by score can reduce gradient diversity within mini-batches, thereby influencing the model to overfit specific patterns rather than learning generalizable features (Jiang et al., 2014; Yin et al., 2018). Consequently, the local diversity guidance suggests the introduction of diverse samples within each micro-batch while maintaining the overall curriculum progress, thereby maximizing the information entropy of gradient estimations and fostering the learning of generalizable features, ultimately improving model generalization.

## B Related Work

### B.1 Data Efficiency

Data efficiency aims to optimize LLMs training by identifying high-quality subsets from massive corpora, a guidance validated by state-of-the-art LLMs (Dubey et al., 2024; OpenAI; Yang et al., 2025) which demonstrate that data quality often outweighs quantity. Existing methods broadly fall into three categories: deduplication, distribution alignment, and quality-based scoring. SemDeDup (Abbas et al., 2023) and D4 (Tirumala et al., 2023) focus on removing semantic redundancy to enhance diversity. Beyond redundancy, DSIR (Xie et al., 2023) selects subsets that mirror target distributions via importance weighting, while PDS (Gu et al., 2025) evaluates sample utility based on gradient consistency. More recently, fine-grained scoring systems have emerged to assess content quality; for instance, FineWeb-Edu (Penedo et al., 2023) employs classifiers to identify educational content, and QuRating (Wettig et al., 2024) evaluates text across multiple dimensions such as writing style and required expertise. However, a critical inefficiency persists across these methods: while substantial computational resources are invested in deriving these scalar metrics, they are typically utilized solely for a one-off binary selection decision. Once the subset is constructed, this rich scoring information is discarded, and the model trains on the retained data agnostic to the significant variance in their intrinsic quality or difficulty.

### B.2 Data Organization

Data organization explores the strategic sequencing of training samples to optimize model convergence and performance. It is distinguished from data efficiency, which focuses on subset selection. The dominant paradigm is Curriculum Learning (Bengio et al., 2009), which advocates for an easy-to-hard progression. Various metrics have been proposed to quantify this difficulty, such as attention scores (Kim and Lee, 2024) or soft edit distance (Chang et al., 2021). Beyond monotonic sorting,

advanced scheduling strategies have emerged: annealing approaches (Dubey et al., 2024) prioritize high-quality data towards the conclusion of training to refine capabilities, while DELT introduces folding learning strategies to mitigate catastrophic forgetting by periodically exposing the model to the full difficulty spectrum (Dai et al., 2025a). However, research on data organization remains fragmented, particularly in the context of LLMs, lacking guidances to guide the data ordering in model training. To bridge this gap, our work is the first to systematically explore the guidances of data ordering, utilizing multi-dimensional scores derived from data efficiency (e.g., selection). This approach allows us to construct sophisticated curricula that enhance model performance and training stability with almost no additional computational overhead.

# C Additional Experimental Setup

## C.1 Setup for Guidances Analysis

### C.1.1 G1: Boundary Sharpening

**Setup.** To systematically investigate the impact of data distribution at the training boundaries, we use the SEG ordering strategy with diverse configurations (see Sec. 3.1). We first isolate the effects of the initialization and ending phases by setting the segment parameter $L = 2$. Based on the percentile intervals of the sorted scores, we define four variants: SEG(h10) ($\tilde{\mathcal{D}}_1 \in [0, 0.1]$, $\tilde{\mathcal{D}}_2 \in [0.1, 1]$); SEG(l90) ($\tilde{\mathcal{D}}_1 \in [0.1, 1]$, $\tilde{\mathcal{D}}_2 \in [0, 0.1]$); SEG(h90) ($\tilde{\mathcal{D}}_1 \in [0, 0.9]$, $\tilde{\mathcal{D}}_2 \in [0.9, 1]$); and SEG(l10) ($\tilde{\mathcal{D}}_1 \in [0.9, 1]$, $\tilde{\mathcal{D}}_2 \in [0.1, 0.9]$).

Furthermore, to explore the joint influence of start-end distributions, we increase the granularity to $L = 3$. This setup includes the variants: SEG(h10-l10) ($\tilde{\mathcal{D}}_1 \in [0, 0.1]$, $\tilde{\mathcal{D}}_2 \in [0.1, 0.9]$, $\tilde{\mathcal{D}}_3 \in [0.9, 1]$) and its reverse SEG(l10-h10), as well as symmetric configurations SEG(l10-l10) ($\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_3 \in [0.9, 1]$, $\tilde{\mathcal{D}}_2 \in [0, 0.9]$) and SEG(h10-h10) ($\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_3 \in [0, 0.1]$, $\tilde{\mathcal{D}}_2 \in [0.1, 1]$). Visualizations of the data distributions for each setting are provided in the Appendix.

### C.1.2 G2: Cyclic Scheduling

**Setup.** To thoroughly investigate the importance of knowledge review, we compare various configurations of CL and FO (see Sec. 3.2). For FO, we set $L \in \{2, 3, 4, 5, 20, 100\}$, based on optimal performance in preliminary folding-layer experiments.

### C.1.3 G3: Curriculum Continuity

**Setup.** To explore the impact of attribute continuity, we compare FO with its corresponding ZIG variants (see Sec. 3.3). Based on their performance of FO, we set our baselines using $L = 2, 3$ for pre-training and $L = 3, 5$ for SFT. Identical settings are applied to ZIG for a consistent comparison.

### C.1.4 G4: Local Diversity

**Setup.** To evaluate the importance of local diversity, we integrate the JIT strategy (Sec. 3.4) into several data ordering variants that are strictly score-ordered, including CL, FO, ZIG. Table 4 presents the best-performing configuration among various JIT settings (as $w$ varies), while more comprehensive results is provided in the Appendix E.

## C.2 Data

(1) **General data.** To evaluate our method on general pre-training scenarios, we utilize two widely adopted corpora that provide comprehensive intrinsic sample-level scores for data selection. **FineWeb-Edu** (Penedo et al., 2023) is a large-scale dataset comprising 1.3T tokens sourced from CommonCrawl (Wenzek et al., 2020). It employs a specialized classifier, distilled from annotations generated by Llama-3-70B, to assign an educational quality score (on a scale of 0 to 5) to each web page, prioritizing content with high knowledge density and logical reasoning. **QuRatedPajama** (Wettig et al., 2024) consists of 260B tokens derived from the CommonCrawl corpus. It distinguishes itself by offering fine-grained quality ratings across four specific dimensions: educational value, writing style, factual content, and required expertise. For the main experiments (Sec. 5.2 and 5.3), we randomly sample 1B tokens from the original corpus, while for the scaling experiments (Sec. 5.4), we sample 50B tokens. Due to space limits, the main text focuses on FineWeb-Edu results while full QuRatedPajama data is deferred to the Appendix.

**Math data.** We utilize **DeepMath-103K** (He et al., 2025) to assess mathematical reasoning capabilities. This dataset comprises approximately 103,000 instruction-tuning samples, constructed by augmenting standard benchmarks (e.g., GSM8K and MATH) with diverse reasoning paths using CoT strategies. It provides a spectrum of difficulty based on the number of reasoning steps and logical complexity.

**Code Data.** For programming tasks, we employ **OpenCodeInstruct** (Ahmad et al., 2025), a

large-scale code instruction dataset synthesized to enhance code generation and execution. It evolves seed instructions into a broad range of algorithmic challenges, where samples are implicitly graded by their algorithmic complexity and execution-based correctness.

### C.3 Model

For pre-training on general data, we adopt the same model architecture as Mistral (Jiang et al., 2023), pre-training variants with 160M, 470M, 1B, and 1.7B parameters. Model configurations follow (Gu et al., 2025), with maximal learning rates following (Brown et al., 2020) detailed in Table 8. For the main experiments (Sec. 5.2 and 5.3), we apply the model with 160M parameters, while for the scaling experiments (Sec. 5.4), we apply models with 160M, 470M, 1B, and 1.7B parameters.

For the SFT stage, we initialize the model with Qwen3-1.7B-Base[1] and conduct full-parameter fine-tuning.

| Model Size | $d_{\text{model}}$ | $d_{\text{FFN}}$ | $n_{\text{layers}}$ | $n_{\text{head}}$ | $d_{\text{head}}$ | learning rate |
|---|---|---|---|---|---|---|
| 160M | 768 | 3,072 | 12 | 12 | 64 | $6 \times 10^{-4}$ |
| 470M | 1,024 | 4,096 | 24 | 16 | 64 | $3 \times 10^{-4}$ |
| 1B | 1,536 | 6,144 | 24 | 16 | 96 | $2.5 \times 10^{-4}$ |
| 1.7B | 2,048 | 8,192 | 24 | 16 | 128 | $2 \times 10^{-4}$ |

Table 8: Model configurations and corresponding learning rates.

### C.4 Training Configuration

For pre-training on general data, all models are trained with a batch size of 256 and a maximum input sequence length of 1,024 for one epoch. The AdamW optimizer (Loshchilov and Hutter, 2019) is paired with a cosine learning rate scheduler. The scheduler includes a warm-up phase for the first 2,000 steps, after which the learning rate decays to 10% of its peak value. The model architecture and corresponding learning rates are summarized in Table 8.

For SFT on mathematics and code datasets, we conduct our experiments using the Llama-Factory framework[2]. The Qwen3-1.7B-Base model is trained using full parameter fine-tuning with a per-device batch size of 4 and 8 gradient accumulation steps for 3 epochs. The maximum input sequence length is set to 2,048, and sequence packing is enabled to improve computational efficiency. A cosine learning rate scheduler is employed with a peak learning rate of $2.0 \times 10^{-5}$. The scheduler includes a warm-up phase for the first 10% of the training steps. All training is conducted in bfloat16 precision.

**Compute Resources.** For the pre-training stage, we train the 160M model on 1B-token datasets using a single NVIDIA A100 GPU. For experiments with the 160M, 470M, 1B, and 1.7B models on 50B-tokens data, we utilize 8*A100 GPUs. For the SFT stage, all experiments are conducted on 4*A100 GPUs.

### C.5 Evaluation

We evaluate the LMs' 0-shot accuracy on the downstream test datasets used in OLMo (Groeneveld et al., 2024). We also report the LM's language modeling loss on a subset of DCLM (Li et al., 2024a). For mathematics, we report results on AIME 2024 and 2025 (AIME, 2025) in the main text, with MATH-500 (Hendrycks et al., 2021) and Minerva Math (Lewkowycz et al., 2022) in the Appendix E. For code, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).

For general scenarios, we assess the trained models on a range of standard natural language understanding and reasoning benchmarks, including Hellaswag (HS; Zellers et al., 2019), Winogrande (Wino; Levesque et al., 2012), LAMBADA (LAMB; Paperno et al., 2016), OpenbookQA (OBQA; Mihaylov et al., 2018), ARC-easy/challenge (ARC-e/c; Clark et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and BoolQ (Clark et al., 2019). We also evaluate the language modeling loss on a subset of DCLM (Li et al., 2024a), a high-quality curated dataset, to confirm that models trained on organized $D_{\text{ord}}$ preserve diversity and long-tail knowledge. For the general benchmarks, the tasks are framed as multiple-choice questions, where the model selects the correct answer by minimizing the normalized loss across all candidate options (`acc_norm`).

For domain-specific tasks, we assess the models on mathematical reasoning and code generation benchmarks. For mathematics, we report results on AIME 2024 and 2025 (AIME, 2025) in the main text, with MATH-500 (Hendrycks et al., 2021) and Minerva Math (Lewkowycz et al., 2022) deferred to the Appendix. For code, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).

For programming benchmarks, we evaluate the fine-tuned model's code generation capabilities on

[1] https://huggingface.co/Qwen/Qwen3-1.7B-Base
[2] https://github.com/hiyouga/LlamaFactory

the HumanEval and MBPP benchmarks using their instruction-based variants. All inputs are formatted using the standard chat template to simulate user-assistant interactions. For HumanEval, we employ a zero-shot setting where the model is prompted to complete a function body based on its signature and docstring. The maximum generation length is set to 1,024 tokens. For MBPP, we adopt a 3-shot setting (`num_fewshot=3`), where the exemplars are provided as multi-turn dialogue history to facilitate in-context learning. The maximum generation length for MBPP is constrained to 256 tokens. For both benchmarks, we utilize greedy decoding (`do_sample=False`) and report the `Pass@1` metric to assess performance.

For the mathematical reasoning benchmarks, we adopt a process-generation, answer-evaluation paradigm. In this setup, the model is prompted with the raw problem text alongside four exemplar demonstrations (4-shot) that contain complete Chain-of-Thought reasoning, and is required to generate the full reasoning process before producing the final answer. We employ a stochastic decoding strategy with a temperature set to 0.7 and top-p set to 0.8 to encourage diverse reasoning paths. During evaluation, only the final answer is extracted and normalized for scoring. To rigorously evaluate performance, we apply dataset-specific configurations: for the competition-level AIME 2024 and AIME 2025, we set the maximum generation length to 2048 tokens and generate k=64 independent samples per problem; for MATH-500 and Minerva Math, we set the maximum generation length to 1024 tokens and generate k=4 samples. The final performance is reported using the `avg@k` metric, which quantifies the model's accuracy across the sampled answers after standard normalization, reflecting the model's consistent performance over multiple reasoning attempts.

## D Test Loss Extrapolation with the Scaling Law

We extrapolate the test losses on the DCLM corpus (Li et al., 2024a) of the conventionally trained and PDS-trained LMs with the Scaling Law (Hoffmann et al., 2022; Kaplan et al., 2020). Following Hoffmann et al. (2022), we consider the scaling law with the following form:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}, \tag{5}$$

where $N$ is the model parameters, $D$ is the number of trained tokens, and $A$, $B$, $E$, $\alpha$, $\beta$ are constants. We obtain these constants by minimizing the Huber loss (Huber, 1992):

$$\min_{a,b,e,\alpha,\beta} \sum_{(N_i,D_i,L_i)} \text{Huber}_{\delta}(\text{LSE}(a-\alpha\log N_i, b-\beta\log D_i, e)-\log L_i), \tag{6}$$

where $\text{LSE}(\cdot)$ is the log-sum-exp operation. The loss is summed over all $(N_i, D_i, L_i)$ tuples, which are obtained by the test losses of 160M, 470M, 1B, and 1.7B LM during training from 0B to 50B tokens. We record the losses every 2.5B tokens, resulting in a total $4 \times 50B/2.5B = 80$ tuples like $(N_i, D_i, L_i)$. After solving $a$, $b$, and $e$ from Eq. 6 , we have $A = \exp(a)$, $B = \exp(b)$, and $E = \exp(e)$.

Hoffmann et al. (2022) optimizes Eq. 6 using the LBFGS algorithm (Liu and Nocedal, 1989). However, we found this algorithm sensitive to the initialization of the parameters to be optimized. Therefore, we apply a two-stage optimization. Specifically, we first fit the following data scaling curves for $N$ =160M, 470M, 1B, and 1.7B with non-linear least squares from `scipy.optimize.curve_fit`[3], which is much more robust to the initialization:

$$L(D) = E'(N) + \frac{B_0(N)}{D^{\beta_0(N)}}, \tag{7}$$

where $E'(N)$, $B_0(N)$ and $\beta_0(N)$ are the fitted parameters. Then, we fit the following model size scaling curve:

$$E' = E_0 + \frac{A_0}{N^{\alpha_0}}. \tag{8}$$

We use the constants from Eq. 8 and the average constants from Eq. 7 to compute the initialization for the LBFGS algorithm:

$$\begin{aligned} a_0 &= \log A_0, \\ b_0 &= \log \frac{B_0(160\text{M}) + B_0(470\text{M}) + B_0(1\text{B}) + B_0(1.7\text{B})}{4}, \\ \alpha_0 &= \alpha_0, \\ \beta_0 &= \frac{\beta_0(160\text{M}) + \beta_0(470\text{M}) + \beta_0(1\text{B}) + \beta_0(1.7\text{B})}{4}, \\ e_0 &= \log E_0, \end{aligned} \tag{9}$$

where $a_0, b_0, \alpha_0, \beta_0, e_0$ are the parameter initialization for the LFBGS algorithm to optimize Eq. 6. We set $\delta = 1 \times 10^{-3}$ and learning rate to 0.05 when running LFBGS and obtain the constants in Table 9. We use these constants and Eq. 5 to compute the predicted loss in Table 7.

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

Table 9: Scaling law constants by fitting the test losses on the DCLM corpus. $E$ represents the irreducible loss inherent to the dataset.

| | $A$ | $B$ | $\alpha$ | $\beta$ | $E$ |
|---|---|---|---|---|---|
| Random | $4.82 \times 10^2$ | $5.12 \times 10^3$ | 0.354 | 0.295 | 1.693 |
| STR | $4.28 \times 10^2$ | $4.55 \times 10^3$ | 0.359 | 0.299 | 1.693 |
| SAW | $4.15 \times 10^2$ | $4.38 \times 10^3$ | 0.361 | 0.302 | 1.693 |

Table 10: Correlations ($R^2$) of fitting the scaling curves.

| | N=160M | N=470M | N=1B | N=1.7B | D=$50\times10^9$ |
|---|---|---|---|---|---|
| Random | 0.992 | 0.995 | 0.998 | 0.997 | 0.999 |
| STR | 0.993 | 0.996 | 0.998 | 0.997 | 0.999 |
| SAW | 0.994 | 0.996 | 0.999 | 0.998 | 0.999 |

**Goodness of Fit.** We evaluate the goodness of fit of the scaling curves with respect to the training token size $D$ and model size $N$ respectively, by computing the correlation coefficient $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$, where $y_i$ is the ground truth value and $\hat{y}_i$ is the prediction.

Regarding the training token size, to make the original problem a linear regression problem, we convert Eq. 5 into

$$\log\left(L(N, D) - E - \frac{A}{N^\alpha}\right) = \log B - \beta \log D. \quad (10)$$

Then, we consider $\log\left(l_i - E - \frac{A}{N^\alpha}\right)$ as the ground truth value for regression, where $l_i$ is the observed loss, and $\log B - \beta \log D_i$ as the prediction. For each $N \in [160M, 470M, 1B, 1.7B]$ we compute an $R^2$ respectively. Similarly, regarding the model size, we convert Eq. 5 into

$$\log\left(L(N, D) - E - \frac{B}{D^\beta}\right) = \log A - \alpha \log N, \quad (11)$$

to compute the corresponding $R^2$ that measures its goodness of fit. For simplicity, we only consider the models at the end of training, where $D = 50 \times 10^9$. The results in Table 10 show that the correlations are sufficiently high, suggesting the scaling curve fits the impact from both the data and model sizes very well.

## E Results on Additional Benchmarks

In this section, we present additional experimental results to further validate our approach.For each guidance setting: (1) we conduct experiments on QuRatedPajama. (2) we extend our evaluation to MATH-500 and Minerva Math. Note that the lower performance on AIME is primarily due to the inherent capacity constraints of the Qwen3-1.7B model.

Tables 11, 12, and 13 provide expanded results for G1, G2, and G3. Table 14 details additional findings for cross-guidance strategies. Moreover, Figure 8 illustrates the performance sensitivity of JIT (G4) across various methods under different window sizes $s$.

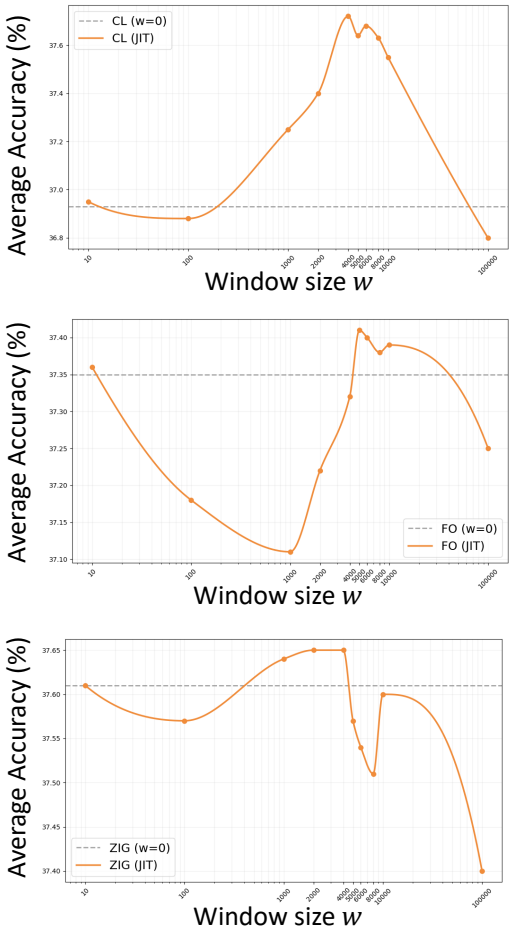


Figure 8: Impact of the JIT window size $w$ across different strategies.

Table 11: Performance comparison of SEG variants and baselines to evaluate G1.

| | **Pre-training** (QuRatedPajama) | | | | | | | | | **SFT** (DeepMath-103K) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | MATH500 | MinervaMath | Avg. |
| Random | 21.08 ± 0.15 | 33.63 ± 0.20 | 27.50 ± 0.14 | 14.26 ± 0.10 | 25.80 ± 0.22 | 55.66 ± 0.45 | 55.50 ± 0.40 | 51.38 ± 0.42 | 35.60 ± 0.07 | 45.65 ± 0.62 | 9.28 ± 0.45 | 27.47 ± 0.12 |
| SEG(h10) | 21.76 | 33.08 | 27.04 | **15.99** | 25.00 | 55.55 | 53.80 | **52.09** | 35.54 | 46.20 | **12.50** | 29.35 |
| SEG(h90) | 21.59 | 30.72 | 27.52 | 12.73 | 23.20 | 55.06 | 53.30 | 49.57 | 34.21 | 47.80 | 11.21 | 29.51 |
| SEG(l10) | 20.56 | 33.42 | **27.81** | 15.89 | 26.00 | **55.82** | **58.80** | 50.36 | 36.08 | **48.55** | 11.12 | **29.84** |
| SEG(l90) | **23.29** | **36.07** | 27.29 | 13.47 | **28.00** | **55.82** | 55.80 | 51.93 | **36.46** | 48.15 | 11.12 | 29.64 |
| SEG(h10-l10) | **22.44** | 30.51 | 27.29 | 13.25 | 24.40 | 54.13 | 54.50 | 49.72 | 34.53 | 44.25 | 9.10 | 26.67 |
| SEG(l10-l10) | 21.67 | 30.43 | 27.33 | 14.21 | 23.60 | 55.44 | 53.50 | 51.46 | 34.70 | 45.65 | 10.66 | 28.16 |
| SEG(l10-h10) | 22.01 | **36.57** | **27.71** | 14.09 | 25.20 | 55.88 | 59.50 | 51.30 | 36.53 | **49.45** | 11.58 | **30.52** |
| SEG(h10-h10) | **22.44** | 35.31 | 27.54 | **14.28** | **26.00** | **57.07** | **60.20** | **51.78** | **36.83** | 46.35 | **13.24** | 29.79 |

Table 12: Performance comparison of FO variants and baselines to evaluate G2.

| | **Pre-training** (QuRatedPajama) | | | | | | | | | **SFT** (DeepMath-103K) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | MATH500 | MinervaMath | Avg. |
| Random | 21.08 ± 0.18 | 33.63 ± 0.23 | 27.50 ± 0.15 | 14.26 ± 0.12 | 25.80 ± 0.20 | 55.66 ± 0.51 | 55.50 ± 0.48 | 51.38 ± 0.04 | 35.60 ± 0.08 | 45.65 ± 0.62 | 9.28 ± 0.45 | 27.47 ± 0.12 |
| CL | 23.63 | **36.99** | 27.63 | 9.90 | 27.00 | 55.22 | 58.40 | 50.20 | 36.12 | 45.70 | 10.48 | 28.09 |
| FO-2 | 23.29 | 36.57 | **27.96** | 11.14 | 27.00 | 56.04 | **59.60** | 51.38 | **36.62** | 47.35 | 9.56 | 28.45 |
| FO-3 | 23.12 | 35.77 | 27.71 | 11.78 | 25.40 | **56.31** | 57.20 | 50.04 | 35.92 | **50.25** | **13.13** | **31.69** |
| FO-4 | 22.18 | 34.89 | 27.51 | 10.87 | **27.40** | 55.98 | 58.30 | **52.25** | 36.17 | 49.20 | 12.04 | 30.62 |
| FO-5 | **23.72** | 34.18 | 27.72 | 11.95 | 25.80 | 54.13 | 56.10 | 52.17 | 35.72 | 44.55 | 10.48 | 27.51 |
| FO-20 | 21.42 | 33.92 | 27.28 | 15.76 | 25.00 | 56.26 | 55.50 | 49.17 | 35.54 | 46.00 | 10.29 | 28.15 |
| FO-100 | 21.25 | 34.05 | 27.78 | **16.20** | 26.20 | 56.04 | 55.70 | 50.51 | 35.97 | 45.05 | 11.86 | 28.45 |

Table 13: Performance comparison of ZIG variants and baselines to evaluate G3.

| | **Pre-training** (QuRatedPajama) | | | | | | | | | **SFT** (DeepMath-103K) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | MATH500 | MinervaMath | Avg. |
| Random | 21.08 ± 0.18 | 33.63 ± 0.23 | 27.50 ± 0.15 | 14.26 ± 0.12 | 25.80 ± 0.20 | 55.66 ± 0.51 | 55.50 ± 0.48 | 51.38 ± 0.04 | 35.60 ± 0.08 | 45.65 ± 0.62 | 9.28 ± 0.45 | 27.47 ± 0.12 |
| FO-2 (or 4) | 23.29 | 36.57 | 27.96 | 11.14 | 27.00 | 56.04 | **59.60** | 51.38 | 36.62 | 49.20 | 12.04 | 30.62 |
| ZIG-2 (or 4) | **23.35** | **36.73** | **29.21** | **11.52** | **27.40** | **56.60** | 58.10 | **52.07** | **36.87** | **49.80** | **12.83** | **31.32** |
| FO-3 | 23.12 | 35.77 | 27.71 | **11.78** | 25.40 | 56.31 | **57.20** | 50.04 | 35.92 | 50.25 | **13.13** | **31.69** |
| ZIG-3 | **23.44** | **36.01** | **27.96** | 11.64 | **26.80** | **56.77** | 56.70 | **52.02** | **36.42** | **50.65** | 12.57 | 31.61 |

Table 14: Performance comparison of cross-guidance strategies (STR and SAW) against other strategies.

| | **Guidances** | | | | **Pre-training** (QuRatedPajama) | | | | | | | | | **SFT** (DeepMath-103K) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | G1 | G2 | G3 | G4 | ARC-c | ARC-e | HS | LAMB | OBQA | PIQA | SciQ | Wino | Avg. | MATH500 | MinervaMath | Avg. |
| Random | - | ✓ | - | ✓ | 21.08 | 33.63 | 27.50 | **14.26** | 25.80 | 55.66 | 55.50 | 51.38 | 35.60 | 45.65 | 9.28 | 27.47 |
| CL (Bengio et al., 2009) | ✓ | - | ✓ | - | 23.63 | 36.99 | 27.63 | 9.90 | 27.00 | 55.22 | 58.40 | 50.20 | 36.12 | 45.70 | 10.48 | 28.09 |
| DELT (Dai et al., 2025a) | ✓ | ✓ | - | - | 23.29 | 36.57 | 27.96 | 11.14 | 27.00 | 56.04 | **59.60** | 51.38 | 36.62 | 49.20 | 12.04 | 30.62 |
| STR (Ours) | ✓ | ✓ | - | ✓ | **25.13** | 37.17 | 28.13 | 12.25 | **27.30** | **56.07** | 57.30 | 52.00 | **36.92** | 54.90 | 11.31 | 33.11 |
| SAW (Ours) | ✓ | ✓ | ✓ | ✓ | 24.49 | **37.34** | **28.55** | 12.28 | 27.20 | 56.04 | 56.40 | **52.14** | 36.81 | **55.10** | **13.42** | **34.26** |