Fine-Grained Data Ordering Improves Fine-Tuning for Large Language Models

Xiaomeng Hu[1], **Yixuan Tang**[2], **Haoze Li**[1], **Hao chen**[1], **Qi Zhang**[1], **Zhanming Shen**[1], **Yiming Zhang**[1], **Haobo Wang**[1], **Junbo Zhao**[1]
[1]ZheJiang University
[2]National University of Singapore
{xm.hu,j.zhao}@zju.edu.cn

Abstract

With the rapid progress of large language models (LLMs), aligning a general-purpose model with downstream tasks through fine-tuning has become a central research focus. Selecting only high-quality examples for training has been shown to be one of the most effective ways to improve fine-tuning performance. However, prior work concentrates almost exclusively on data preprocessing: filtering and cleaning data before training begins. While the order and composition of training data during training have received little fine-grained attention. To fill this gap, our work proposed ***F**ine-Grained **O**rder Fine-**T**uning*, a fine-grained scheduling method of data order in epochs. Drawing on curriculum-learning principles, FOT defines data difficulty based on the relevance between the data and the model, and then performs dynamic scheduling of the training order in each epoch according to the difficulty. On both large-scale continued pre-training and small-scale supervised fine-tuning experiments, FOT has achieved an average 2.4% improvement over baselines. Our study offers a new perspective on data governance in the fine-tuning phase.

1 Introduction

In recent years, large language models (LLMs) have advanced rapidly. General-purpose LLMs have demonstrated near-human-level capabilities in language understanding and reasoning (Achiam et al., 2023; Grattafiori et al., 2024; Guo et al., 2025; Yang et al., 2024). To bridge the gap between broad pretraining and domain-specific applications, fine-tuning and alignment techniques—such as continued pretraining and supervised fine-tuning—have become essential. These approaches not only strengthen LLMs' domain adaptation, but also substantially improve their reasoning performance. (Wei et al., 2022; Sanh et al., 2022; Xie et al., 2024; Guo et al., 2025). As fine-tuning techniques continue to advance, research into fine-tuning data itself has also emerged as a major focus. Recent studies have demonstrated that the quality of fine-tuning data significantly impacts model performance (Zhou et al., 2023; Pang et al., 2025a). Consequently, data cleaning, filtering, and subset selection have become critical components in enhancing the effectiveness of fine-tuning (Chen et al., 2023). Fine-tuning with only a small amount of high-quality data can achieve, or even surpass, the performance of fine-tuning with large-scale data (Zhou et al., 2023).

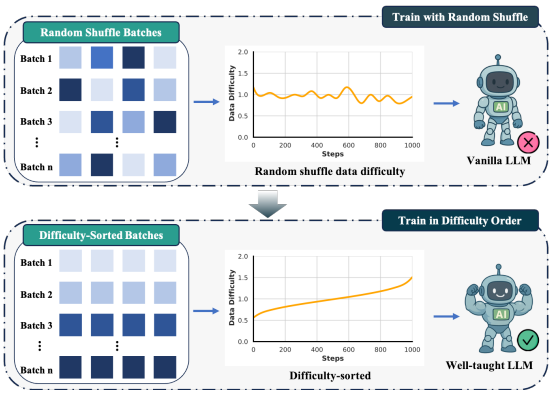


Figure 1: In conventional training setups, data within each epoch is fed into the model in a randomly shuffled order, as shown in the upper part. Such an approach overlooks important information both between data samples and between the data and the model. In contrast, our method applies fine-grained sorting based on data difficulty and feeds samples into the model in a difficulty-aware order, as illustrated in the lower part.

However, existing research on fine-tuning data has predominantly focused on the pre-processing stage, where data are curated and filtered before training. Most existing work focuses solely on the properties of the data itself, such as its relevance to the task, without considering how to leverage the data more effectively during the training process. There has been a lack of fine-grained, systematic study on the ordering and composition of training data during the fine-tuning process.

Findings of the Association for Computational Linguistics: ACL 2026, pages 20406–20418
July 2-7, 2026

For fine-tuning of large language models, the essence is to adapt the model from the pre-training distribution to the target task distribution (Shengyu et al., 2023). Scheduling the training data so that the model gradually transitions from the original distribution to the target distribution has been shown to better promote convergence toward the target (Abnar et al., 2021; Xu et al., 2021; Weinshall et al., 2018). Recently, many approaches in the reinforcement learning stage of large language models have leveraged curriculum learning principles to schedule training tasks (Parashar et al., 2025; Jiang et al., 2025). However, most existing fine-tuning work presents training data in a randomly shuffled order. Because the data is unordered, they cannot achieve the desired progressive training effect. To address this issue, we conduct a fine-grained investigation into the role of training data order during LLM fine-tuning and propose a curriculum-inspired approach named ***F**ine-Grained **O**rder Fine-**T**uning*, which dynamically adjusts the sequence of training examples from easier to harder, thereby approximately realizing a progressive transition from the original distribution to the target distribution. Concretely, for each candidate training sample, **FOT** estimates its **difficulty** based on two criteria: **(1) the cross-entropy loss of the model in that sample, and (2) the probability that the sample's content was seen during the original pre-training.** After obtaining the difficulty scores of data samples, **FOT** draws on the principles of curriculum learning to schedule the training order for each epoch: **during the early stages of finetuning, we train samples from easy to difficult, while in later stages, we re-shuffle the data in random order to avoid overfitting.**

To rigorously assess the effectiveness of our method, we conducted comprehensive experiments. For domain-specific data such as code and mathematics, we carried out not only large-scale continued pre-training, but also small-scale supervised fine-tuning, and evaluated performance on corresponding benchmarks. The experimental results consistently show that our approach achieves improvements over baseline methods across a range of fine-tuning settings and model scales. Furthermore, by visualizing the changes in data distributions and gradients throughout training, and through various ablation studies such as out-of-distribution evaluations, we demonstrate the effectiveness of our approach.

In summary, our contributions are as follows:

- A pioneering fine-grained investigation into the training data order during the fine-tuning phase. We offer a brand-new perspective on data governance during the fine-tuning stage.
- Comprehensive Continued Pre-Training and Supervised Fine-Tuning experiments demonstrate superior performance over baselines across different training scales, model sizes, and tasks.

2 Related Work

2.1 Data Centric Fine-tuning

Recent years, a growing body of work showed that the composition and quality of supervision data dominate the outcome of LLM alignment. LIMA (Zhou et al., 2023) demonstrates that merely 1K carefully curated instruction–response pairs can rival or surpass proprietary systems, challenging the premise that more data is always better. Following the same philosophy, AlpaGasus (Chen et al., 2023) filters the 52K Alpaca corpus down to 9K GPT-4-verified examples and matches 90% of its teacher's performance while reducing training cost 5.7×.

Beyond example-level filtering, more granular data cleaning methods have also become a focus of attention. RHO-1 (Lin et al., 2024) proposed selective language modeling(SLM), which only calculates important tokens when calculating loss. Pang et al. (2025b) introduce Token Cleaning, pruning low-influence tokens inside good samples and reporting consistent SFT gains across tasks.

2.2 Curriculum Learning and Training-order Effects for LLMs

Curriculum learning (CL) advocates presenting training instances from easy to hard, a principle formalised by Bengio et al. (2009) and its self-paced variant by Kumar et al. (2010). Early NLP applications, such as competence-based schedules for MT (Platanios et al., 2019) and difficulty-aware pointer generator readers (Tay et al., 2019), showed faster convergence and better generalization.

Recent work adapts Curriculum Learning to large language models. IT2ACL (Huang and Xiong, 2024) builds a two-phase, loss-driven syllabus that feeds LLMs increasingly difficult instruction clusters and reports consistent gains across 16 tasks. Ranaldi et al. (2024) study whether the training order matters in multilingual settings, proposing curriculum learning over languages to improve

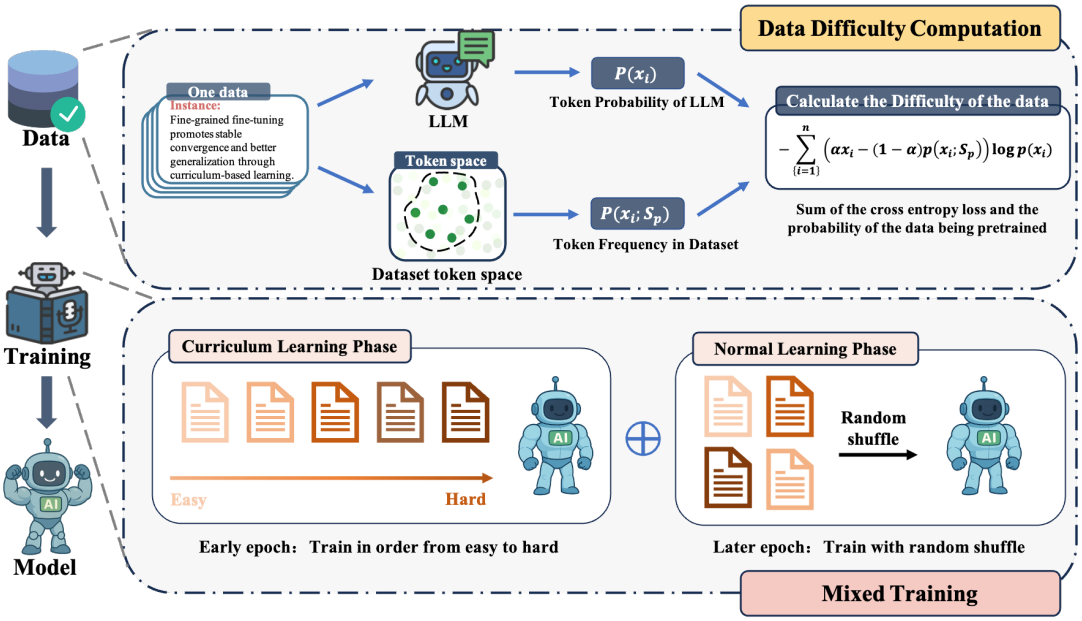


Figure 2: Overview of FOT framework. On the data side, FOT first feeds each sample into the model to obtain the model-assigned probability for every token $P(x)$, while simultaneously computing the corpus-level token-frequency distribution $P(x_i; S_p)$; these two quantities are then merged via Equation $-\sum_{i=1}^{n}(\alpha x_i-(1-\alpha)p(x_i, S_p))\log p(x_i)$ to give each sample a difficulty score, defined as a weighted sum of the sample's cross-entropy loss and the prior probability the token already trained after pre-training. On the training side, FOT splits training into two phases: during the early curriculum phase the model receives examples in ascending order of difficulty—moving from easy to hard—whereas in the later phase the same data are randomly shuffled before presentation, a strategy that preserves the curriculum's benefits while limiting over-fitting.

cross-lingual transfer. Zhang et al. (2025c) propose replacing random sampling with difficulty-aware curricula for more efficient language model pre-training. Zhang et al. (2025a) improve LLM reasoning via adaptive difficulty curriculum learning combined with expert-guided self-reformulation. In the alignment setting, 2D-Curri-DPO (Li and Zhang, 2025) extends Curriculum-DPO with a two-dimensional difficulty axis (prompt complexity × preference clarity) and achieves state-of-the-art MT-Bench scores while remaining RL-free.

Beyond curricula, some papers probe how the micro-level sequence of updates influences fine-tuning stability. Kim and Lee (2024) demonstrates that re-ordering just 4% of a 180K instruction set by easiness yields small but consistent wins, yet their study still treats ordering in coarse buckets. However, fine-grained, epoch-level scheduling for LLM fine-tuning remains largely unexplored, and there is a lack of systematic research in this area.

3 Method

3.1 Preliminary

Consider the fine-tuning dataset S_f with its data $x = \{x_1, x_2, \cdots, x_n\} \in S_f$, where n is the sequence length, and S_f follows a distribution D_f. For simplicity, we treat each data point holistically without distinguishing between input and output components. Given a pre-trained LLM, its pre-training dataset S_p also follows a distribution D_p. When x is fed into the LLM, the corresponding token-level probabilities are $P(x_1), P(x_2), \cdots, P(x_n)$.

3.2 Fine-grained scheduling of the training data order

For the data within an epoch, the existing training procedure partitions all data into m batches $B_1, B_2, \ldots, B_m$. Assume the data in each batch follows a distribution D_i, and training proceeds in the order $1 \to m$. Let Δ denote the distance between two probability distributions. Intuitively, an ideal training order should satisfy the following conditions:

$$\begin{aligned}\Delta(D_{i+1}, D_p) &\geq \Delta(D_i, D_p)\\ \Delta(D_{i+1}, D_f) &\leq \Delta(D_i, D_f)\end{aligned} \quad (1)$$

That is, over the course of training, we gradually shift from the original distribution toward the target

distribution. Initially, the training data is closer to the center of the pretraining distribution; as training proceeds, it moves toward the center of the target distribution. This gradual migration smooths the gradients during convergence and better facilitates the model's fit to the target domain (Abnar et al., 2021; Xu et al., 2021; Weinshall et al., 2018).

However, most existing methods train in a random order, as shown in the upper panel of Figure 1, which prevents them from achieving the desired training sequence. Therefore, it is worthwhile to plan the training order at a finer granularity. Since it is difficult to precisely estimate the distribution of text data, inspired by curriculum-learning principles, data difficulty emerges as a key criterion for organizing the training sequence. Easy examples are more likely to be close to the original data distribution D_p, whereas hard examples are more likely to deviate from the original data and be closer to the target data distribution D_f. Therefore, training from easy to hard, as shown in the lower panel of Figure 1, can approximately achieve the ideal training order described in Equation 1. Section 5.1 also corroborates this through visualizations.

The next two sections formally define data difficulty and present our detailed procedure for scheduling the training order.

3.3 Definition of Data Difficulty

Given a data sample x and a pre-trained LLM, the difficulty of this data for the model can be determined by the following two aspects:

- **Cross-entropy loss of the data with respect to the model.** When the data is fed into the model, a higher output loss indicates greater difficulty of the data, and vice versa. We define this component as

$$D_{ce} = -\sum_{i}^{n} x_i \log P(x_i) \qquad (2)$$

- **The probability of the data being trained in the model's pre-training.** If a data sample is more likely to have been encountered during pre-training, it is considered easier for the model; conversely, less probable samples are deemed more challenging. Drawing inspiration from training data detection methodologies, we adapt the DC-PDD (Zhang et al., 2024) approach to formally define this component as:

$$D_{pd} = \sum_{i}^{n} P(x_i; S_p) \log P(x_i) \qquad (3)$$

where $P(x_i; S_p) = \frac{\text{count}(x_i)}{N_p}$ represents the token frequency distribution, $\text{count}(x_i)$ denotes the occurrence count of the token x_i in the dataset and N_p is the total number of tokens in S_p. Since the original pretraining dataset is unavailable, we follow the approach adopted in the original paper and use a large-scale pretraining corpus (RedPajama (Weber et al., 2024)) as an approximation. To maintain consistency with the loss metric, we invert the original value proposed in the reference paper, ensuring that higher values correspond to greater difficulty levels.

Algorithm 1 **F**ine-Grained **O**rder Fine-**T**uning(FOT)

Input: The Fine-tuning data $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, The Fine-tuning Dataset S_f with total token number N_p, The Pre-trained Model M. The Function $p_{\text{LLM}}()$ is to get the Probability of the LLM's outputs. Training Epochs e. Hyper-Parameters α, ϵ.

Output: Fine-tuned M_f

```
P(x_i) ← p_LLM()
P(x_i; S_f) ← count(x_i) / N_f
D_ce ← −Σ_i^n x_i log P(x_i)
D_pd ← D_pd = Σ_i^n P(x_i; S_p) log P(x_i)
Diff(x) ← αD_ce + (1 − α)D_pd
Sort S_f from simple to difficult according to Diff(x)
for i ← 1 to e do
    if i < ϵ then
        Train M with sorted S_f
    else
        Train M with random-shuffled S_f
    end if
end for
return M_f
```

Based on the two components above, we formally define the data difficulty as follows:

Definition $\text{Diff}(\mathbf{x})$ *The difficulty of a data sample is computed as the weighted average of its cross-entropy loss and the probability of being covered*

in pre-training.

$$\begin{aligned}\text{Diff(x)} &= \alpha D_{ce} + (1-\alpha) D_{pd} \\ &= -\sum_{i=1}^{n} (\alpha x_i - (1-\alpha) P(x_i; S_p)) \log P(x_i)\end{aligned} \tag{4}$$

3.4 Training Sequence Scheduling

After annotating the difficulty scores for the training dataset, we subsequently plan the training sequence by following conventional curriculum learning methodologies. The complete fine-tuning process is divided into two phases:

1. **Curriculum Learning Phase.** In the early epochs of training, we sort the data from easy to difficult and then feed it into the training model in the sorted order. Within each epoch, simpler samples are trained first, followed by more challenging ones. To maintain a degree of randomness, we designed a block-wise shuffling method: the sorted data is divided into several blocks, and shuffling is performed within each block. This approach not only preserves the randomness of the data but also ensures that the overall training process progresses from easy to difficult.

2. **Normal Training Phase**. In later epochs of training, we randomly shuffle the entire dataset before training to prevent the model from overfitting.

3.5 Method Overview

Our approach, named ***F**ine-Grained **O**rder Fine-**T**uning (FOT)*, is described in Algorithm 1. Given a fine-tuning dataset S_f, **FOT** first feeds it through the model to obtain the predicted probabilities $P(x_1), P(x_2), \cdots P(x_n)$. We then compute a difficulty score $\text{Diff}(x)$ for each sample using Equation 4. After obtaining these scores, the entire dataset is sorted in ascending order—from the easiest to the hardest examples. During training, **FOT** carefully schedules every epoch. In the first epoch, the model is trained with the data in the pre-sorted order. In the second epoch, we switch to the conventional strategy of randomly shuffling the data before each pass. Figure 2 gives a complete overview of the workflow of our method.

4 Experiment

The experiments in this work consist of two parts: continued pre-training and supervised fine-tuning. This section details the experimental setup and presents the results. Further experimental details can be found in the apeendix.

4.1 Experiment Setup

Datasets. In the continued pre-training phase, the CodeParrot dataset was selected, from which a subset of data was extracted as the code-focused dataset. Additionally, following the practices of previous works in continued pre-training, the RedPajama (Weber et al., 2024) dataset was incorporated to mitigate catastrophic forgetting, with a subset of data serving as general-purpose content. The two datasets were mixed in a 4:1 ratio of code to general data, resulting in a combined dataset of approximately 2 billion tokens. In the SFT phase, two datasets were selected to cover both mathematical and coding domains: MathInstruct and Magicoder, serving as the training sets for the respective tasks.

Models. Due to computational resource constraints, the CPT experiments were conducted solely on the LLama-3.2-1B model, while on the SFT experiments, four base models of varying sizes and types were selected: LLama-3.2-1B, LLama-3.1-8B (Grattafiori et al., 2024), Gemma-3-4B (Team et al., 2025) and Qwen-3-4B (Yang et al., 2025).

Evaluation. To evaluate the model's code-related capabilities, assessments were conducted on several code benchmarks, including HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023), MBPP (Austin et al., 2021), and MBPP+ (Liu et al., 2023) for both CPT and SFT. Furthermore, to assess the extent of catastrophic forgetting, evaluations were performed on general-purpose benchmarks such as MMLU (Hendrycks et al., 2021a) and CMMLU (Li et al., 2024) for CPT. For the math domain in SFT, evaluations are performed on the GSM8K (Austin et al., 2021) and Math (Hendrycks et al., 2021b) benchmarks to comprehensively measure the performance of the model in math tasks. For each dataset, we tested and averaged the results twice.

Training Details. For each experimental setting, the training process consists of two epochs: the first epoch corresponds to Phase 1 as described in Section 3.4, and the second epoch corresponds to Phase 2.

Baselines. To validate the effectiveness of our proposed training data scheduling method, the fol-

Method	General		Code				Avg.	
	MMLU	CMMLU	HEval	HEval+	MBPP	MBPP+		
Random	29.16	26.17	11.59	10.37	26.46	21.96	20.96	0.00
Anti-Curri	28.54	25.35	13.41	12.20	25.40	21.69	21.12	↑0.16
MIX	27.82	24.95	12.80	11.59	**27.51**	21.96	21.11	↑0.15
IT2ACL	29.22	25.35	12.20	11.59	25.66	21.43	20.90	↓0.06
Kim and Lee (2024)	29.43	25.02	12.20	10.98	25.40	21.43	20.57	↓0.39
FOT	**29.57**	**26.66**	**14.63**	**14.01**	27.25	**22.22**	**22.39**	**↑1.43**

Table 1: Performance of our method in CPT compared to other baselines with Llama-3.2-1B in general and code tasks. The best results are highlighted in **bold** and the second best results are highlighted in underline. ↓↑ indicates change relative to the baseline. Our method achieved the best performance on average, outperforming the baseline by 1.4%.

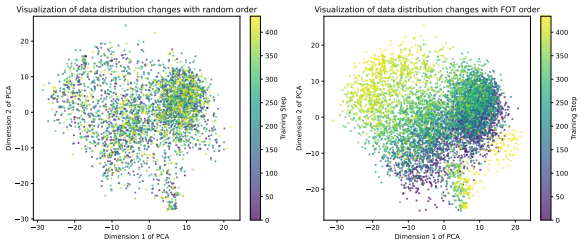
Model	Method	Math		Code				Avg.	
		GSM8K	MATH	HEval	HEval+	MBPP	MBPP+		
Llama-3.2-1B	Random	17.89	3.76	25.61	21.95	19.31	16.40	17.49	0.00
	Anti-Curri	**17.97**	4.52	27.44	24.39	19.84	15.61	18.28	↑0.79
	MIX	16.62	4.00	26.22	23.17	20.63	**17.99**	18.14	↑0.65
	IT2ACL	17.44	4.82	28.66	**25.61**	19.84	16.93	18.86	↑1.37
	Kim and Lee (2024)	17.29	4.12	26.82	23.17	19.04	16.93	17.90	↑0.41
	FOT	16.90	**5.12**	**30.18**	24.39	**20.90**	17.86	**19.23**	**↑1.74**
Llama-3.1-8B	Random	55.95	20.20	67.07	**64.63**	46.83	39.42	49.01	0.00
	Anti-Curri	49.20	21.42	68.29	60.98	50.53	40.74	48.52	↓0.49
	MIX	53.83	19.18	65.24	59.76	48.41	41.27	47.97	↓1.04
	IT2ACL	**58.15**	20.52	63.41	60.37	**50.79**	42.06	49.23	↑0.21
	Kim and Lee (2024)	54.38	20.88	67.07	63.41	49.47	**42.33**	49.61	↑0.60
	FOT	57.28	**22.38**	**68.90**	62.80	50.93	41.93	**50.70**	**↑1.69**
Gemma-3-4B	Random	54.00	20.58	64.63	60.98	58.20	50.00	51.40	0.00
	Anti-Curri	54.21	20.52	65.24	60.98	59.52	51.59	52.06	↑0.66
	MIX	54.74	21.22	67.07	63.41	57.40	50.53	52.38	↑0.98
	IT2ACL	**58.07**	21.44	67.68	64.63	59.79	51.59	53.88	↑2.48
	Kim and Lee (2024)	56.10	21.36	67.68	65.24	58.99	52.91	53.04	↑1.64
	FOT	57.62	**21.80**	**70.12**	**67.68**	**64.29**	**52.91**	**55.74**	**↑4.34**
Qwen-3-4B	Random	60.50	25.92	78.66	73.78	74.60	62.70	62.69	0.00
	Anti-Curri	59.51	25.56	76.83	73.17	74.87	62.70	62.10	↓0.59
	MIX	60.50	26.20	78.05	73.78	74.07	63.49	62.68	↓0.01
	IT2ACL	61.11	**28.68**	**79.88**	74.39	73.02	62.17	63.21	↑0.52
	Kim and Lee (2024)	60.80	26.54	78.05	72.56	74.34	64.02	62.72	↑0.03
	FOT	**62.32**	28.52	79.57	**75.00**	**75.79**	**64.55**	**64.29**	**↑1.60**

Table 2: Performance of our method in SFT compared to other baselines with Llama-3.2-1B, Llama-3.1-8B, Gemma-3-4B and Qwen-3-4B in math and code tasks. For LLama-3.2-1B, FOT achieved the best performance on average, outperforming the baseline by 1.9%. For LLama-3.1-8B, FOT achieved the best performance on average, outperforming the baseline by 1.6%. For Gemma-3-4B, FOT outperformed the baseline by 4.3%, and for Qwen-3-4B, FOT outperformed the baseline by 1.6%.

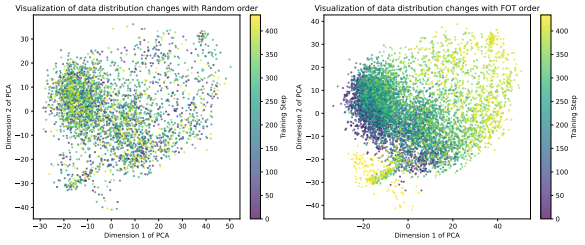
lowing baselines were designed for comparison: **Random**: Data are shuffled in a random order for training. **Anti-Curri**: Data are presented in reverse curriculum order, from the most difficult to the simplest samples. **MIX**: Each 1% subset of the data is composed of a mixture of easy and difficult samples, ensuring that each portion contains both easy and hard examples. **IT2ACL**: Following the approach of IT2ACL (Huang and Xiong, 2024), we take Instruction Prediction Gain as the difficulty metric, that is, how much additional loss can still be reduced in each instruction, and train the model in an easy-to-hard curriculum order. **Kim and Lee (2024)**: Following the approach of Kim and Lee (2024), use the length and attention score as a metric and train the model in order. All of these methods are applied only during the first phase of training described in Section 3.4, while the second phase follows the same training strategy across all methods.

Model	Method	Description	Math		Code				Avg.	
			GSM8K	MATH	HEval	HEval+	MBPP	MBPP+		
Llama-3.2-1B	Random	Only Phase 2	**17.89**	3.76	25.61	21.95	19.31	16.40	17.49	0.00
	Only CL	Only Phase 1	17.82	4.30	24.39	17.68	**22.22**	**19.84**	17.70	↑0.21
	FOT	Phase 1+2	16.90	**5.12**	**30.18**	24.39	**20.90**	17.86	**19.23**	**↑1.74**
Llama-3.1-8B	Random	Only Phase 2	55.95	20.21	67.07	64.63	46.83	39.42	49.01	0.00
	Only CL	Only Phase 1	54.79	**23.52**	46.34	40.85	43.12	37.30	40.99	↓8.02
	FOT	Phase 1+2	57.28	**22.38**	**68.90**	62.80	50.93	41.93	**50.70**	**↑1.69**
Gemma-3-4B	Random	Only Phase 2	54.00	20.58	64.63	60.98	58.20	50.00	51.40	0.00
	Only CL	Only Phase 1	54.98	21.40	67.07	64.63	56.08	50.00	52.36	↑0.96
	FOT	Phase 1+2	**57.62**	**21.80**	**70.12**	**67.68**	**64.29**	**52.91**	**55.74**	**↑4.34**
Qwen-3-4B	Random	Only Phase 2	60.50	25.92	78.66	73.78	74.60	62.70	62.69	0.00
	Only CL	Only Phase 1	52.39	26.04	76.83	71.95	59.52	50.26	56.17	↓6.52
	FOT	Phase 1+2	**62.32**	28.52	79.57	**75.00**	**75.79**	**64.55**	**64.29**	**↑1.60**

Table 3: Ablation Study on only Train by Curriculum Learning. The results show that if we only perform training in the CL (Curriculum Learning) stage, the model improves on certain metrics but exhibits significant drops on others. In the 8B-scale experiments, it even suffers from substantial performance degradation.



(a) Magicoder



(b) MathInstruct

Figure 3: Visualization of data distributions. The left figure shows distributional changes under random ordering, and the right figure shows changes under FOT. As observed, with a random order the overall distribution evolves chaotically, whereas with our method the distribution exhibits a progressive trend throughout training.

4.2 Experiment Results

The experimental results for CPT are presented in Table 1. Among all approaches, our method delivers the best performance. Compared with the baseline, **FOT** achieves an average improvement of around 1.5%. Although certain metrics are best achieved by alternatives such as MIX, **FOT** surpasses all other methods overall.

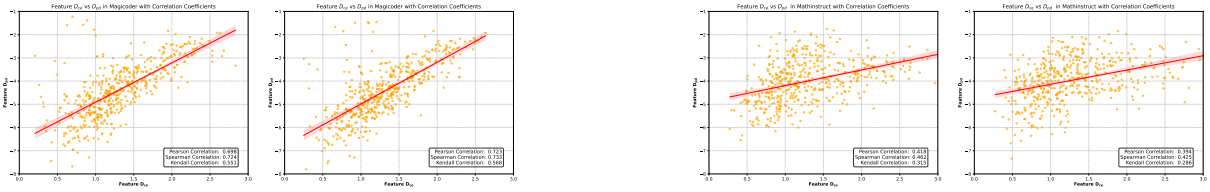
The results of SFT are shown in Table 2. Across all four models, our approach attains state-of-the-art performance, surpassing the baseline by an average of 2.4%. On Gemma-3-4B in particular, the **FOT** method yields a marked gain of 4.5% over the baseline. The remaining models likewise exhibit consistent improvements relative to both the baseline and competing methods. Results on the CPT and SFT benchmarks further substantiate the effectiveness of **FOT**.

5 Analysis

5.1 Visualization of Data Distribution and Gradient

As discussed in Section 3.2, training progressively, from the original pretraining distribution toward the target distribution, better promotes convergence to the target domain. Following curriculum learning principles, we train from easy to hard, which approximates this idealized process. To further verify this, we visualize how the data distribution evolves over training steps within a single epoch. For each example, we take the last-layer hidden state of the pretrained model (e.g., LLaMA-3.2-1B) as its representation, apply PCA for dimension reduction, and visualize the results, as shown in Figure 3.

From the distributional changes, we can see that unlike the random order, our method exhibits a clear gradual transition. In the early training steps, the data distribution is close to one distributional center. Throughout training, data from adjacent steps are also close in distribution; in the later steps, the data distribution approaches another distributional center. Therefore, we validate that our method can approximately simulate the ideal training process described in Equation 1.



(a) Magicoder (b) MathInstruct

Figure 4: The two features exhibit a strong positive correlation, which demonstrates the validity of both metrics.

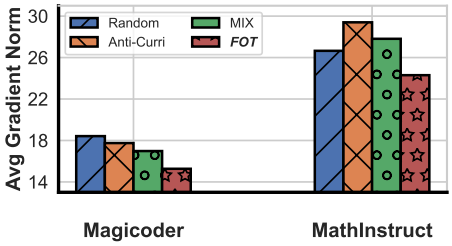


Figure 5: Gradient norm scale during training. FOT achieves the smallest gradient norm compared to other methods, resulting in a smoother optimization process.

Furthermore, we visualize the average gradient norm (before clipping) during training for different methods, as shown in Figure 5. We observe that our method attains the smallest gradient norm, leading to a smoother overall training process and facilitating convergence toward the global optimum.

5.2 Two Phase Analysis

In Section 3.4, we divide the training into two stages, a curriculum learning (CL) stage and a standard training stage. To assess the benefit of this two-stage design, we compare FOT with ablations that retain only one stage. Keeping only the standard stage corresponds to the random baseline setting, while the CL-only ablation trains for every epoch strictly in the easy-to-hard order.

As Table 3 shows, curriculum learning training alone delivers strong performance on MBPP(+), but causes a sharp drop in HumanEval(+). In the 8B-scale experiments, it even suffers from substantial performance degradation. We suspect this reflects over-fitting to parts of the data. To avoid that, we adopt the two-stage training scheme.

5.3 Relationship Between Difficulty Metrics

In Section 3.3, the difficulty measure is composed of two parts: the loss of cross-entropy D_{ce} and the probability D_{pd} that the data are trained before the model training. To investigate the correlation between these two metrics, we visualize both features for every data point by plotting a scatter diagram with D_{ce} on the x-axis and D_{pd} on the y-axis, fitting a straight line to the plot, and then calculating the Spearman and Pearson correlation coefficients between them. A visual analysis of the difficulty metrics is shown in Figure 4. Across both datasets, metrics D_{ce} and D_{pd} exhibit a strong positive correlation, indicating that both effectively capture the underlying data difficulty.

Metric	HEval	HEval+	MBPP	MBPP+	Avg.
Random	64.63	60.98	58.20	50.00	58.45
Only D_{pd}	68.29	66.46	61.64	53.17	62.39
Only D_{ce}	67.68	60.98	59.25	50.26	59.54
FOT	**70.12**	**67.68**	**64.29**	**52.91**	**63.75**

Table 4: Ablation experiments of two difficulty metrics in Gemma-3-4B.

Additionally, we conduct ablation studies on Gemma-3-4B by training the model using only on metric D_{pd} or metric D_{ce}, respectively. The results are shown in Table 4. When using a single difficulty metric, the performance improves to some extent, but using both metrics together yields better results. As a result, we set the coefficient $\alpha = 0.5$ in Equation 4 as the final value.

5.4 Out of Distribution Analysis

To further validate the effectiveness of our method, we devise experiments on out-of-distribution (OOD) data. Specifically, when the model is trained on a code dataset, we evaluate it on a math benchmark; conversely, when it is trained on a math dataset, we test it on a code benchmark. This setup allows us to determine whether different methods experience catastrophic forgetting.

The OOD results are reported in the Appendix B. While **IT2ACL** attains the best score on some models, **FOT** achieves the highest overall OOD performance across all methods, indicating that it effectively mitigates catastrophic forgetting.

6 In context-learning Toy Experiment Demonstration

Dataset	Difficulty	Demonstration Type		
		None	Easy	Hard
Mathinstruct	Easy	0.903	0.844 ↓6.5%	0.899 ↓0.4%
	Hard	1.714	1.640 ↓4.3%	1.544 ↓9.9%
Magicoder	Easy	0.873	0.848 ↓2.9%	0.964 ↑10.4%
	Hard	1.817	1.727 ↓5.0%	1.743 ↓4.1%
CodeParrot	Easy	0.812	0.749 ↓7.8%	0.794 ↓2.2%
	Hard	1.483	1.406 ↓5.1%	1.364 ↓8.0%

Table 5: For all datasets, a larger drop in loss occurs when the demonstrations and test questions share the same difficulty level (i.e., E2E and H2H). When the demonstrations are easy but the questions are hard (E2H), the loss still decreases significantly—indeed, it achieves the best result on Magicoder. In contrast, when the demonstrations are hard but the questions are easy (H2E), the loss even increases.

6.1 In-Context Learning Experiment

Recent studies showed that the essence of in-context learning is that the model performs implicit gradient descent on the demonstration samples (von Oswald et al., 2023; Dai et al., 2023; Zhang et al., 2025b). Therefore, we simulate a simple fine-tuning scenario through in-context learning. For a given dataset D, we annotate the difficulty levels of the data using Equation 4. After annotation, we define the easiest 20% of the data as **easy** samples and the hardest 20% as **hard** samples.

To simulate a training process from easy to hard, we use easy samples as demonstrations and take the questions from hard samples as input, then compute the loss on the answer part. We refer to this input format as "Easy-to-Hard In-Context Learning" (E2H). Conversely, we also experiment with "Hard-to-Easy In-Context Learning" (H2E), where hard samples serve as demonstrations and easy sample questions are used as input, with the loss computed on the answers. For comparison, we also designed Easy-to-Easy in-context learning (E2E) and Hard-to-Hard in-context learning (H2H). Both approaches follow the same design methodology as described above.

6.2 Experimental Details

To align with the main experiments described earlier, we selected the same datasets Magicoder, Mathinstruct and CodeParrot for this study. For each problem instance, we adopted a 5-shot learning approach by randomly sampling five data points (either from easy or hard subsets) to serve as demonstrations. The prompt used for ICL is as follows: *"Here are some examples of questions and answers. Please answer the last question based on these examples: Question: . Answer: ."* In addition, we included a baseline condition in which individual question-answer pairs were fed directly into the model without any demonstrations. To evaluate performance, we computed the cross-entropy loss between the model's outputs and the ground-truth answers as our metric.

6.3 Result Analysis

Our experimental results are presented in Table 5. The key observations are as follows.

Superior Performance of Matched-Difficulty Demonstrations: Across all results, both the easy-to-easy (E2E) and hard-to-hard (H2H) demonstrations achieved the most significant improvements over the baseline. This strongly validates that when data samples share similar difficulty levels, their spatial distributions become more homogeneous, leading to greater reduction in ICL loss. These findings confirm that higher similarity between batch samples in fact enables smoother convergence.

Asymmetric Transfer Effects: The easy-to-hard (E2H) demonstrations showed marked improvement over the baseline, while hard-to-easy (H2E) performance remained virtually identical to the baseline. This striking asymmetry corroborates that for a pre-trained model, training with easier samples first provides positive transfer to harder samples, whereas the reverse sequence offers negligible benefit.

7 Conclusion

This paper proposes ***F**ine-Grained **O**rder Fine-**T**uning (FOT)*, a fine-grained method for scheduling the training order based on data difficulty. Our method schedules the input sequence within each epoch in a fine-grained manner. Experiments on both CPT and SFT show that FOT achieves strong performance. We further demonstrate the effectiveness of FOT through extensive ablation experiments. Our work offers a new perspective on data governance during the fine-tuning stage.

Limitations

Due to computational resource constraints, our experiments were conducted only on models of 1B to 8B parameters, without evaluation on a broader

range of model scales. Moreover, the intuitive impact of the FOT method on model training still requires deeper analysis. In the future, we plan to extend FOT to more model sizes and adopt more interpretable data to better demonstrate its effectiveness.

Acknowledgments

This work is supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China. This work is also supported by ZJU Kunpeng&Ascend Center of Excellence.

References

Samira Abnar, Rianne van den Berg, Golnaz Ghiasi, Mostafa Dehghani, Nal Kalchbrenner, and Hanie Sedghi. 2021. Gradual domain adaptation in the wild: When intermediate distributions are absent. *arXiv preprint arXiv:2106.06080*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

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.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and 1 others. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

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. *Preprint*, arXiv:2107.03374.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *Preprint*, arXiv:2212.10559.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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

Yufei Huang and Deyi Xiong. 2024. It2acl learning easy-to-hard instructions via 2-phase automated curriculum learning for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9405–9421.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. 2025. Vcrl: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*.

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

M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Mengyang Li and Zhong Zhang. 2025. 2d-curri-dpo: Two-dimensional curriculum learning for direct preference optimization. *arXiv preprint arXiv:2504.07856*.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and 1 others. 2024. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Preprint*, arXiv:2305.01210.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Jinlong Pang, Na Di, Zhaowei Zhu, and et al. 2025a. Token cleaning: Fine-grained data selection for llm supervised fine-tuning. *arXiv preprint arXiv:2502.01968*.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. 2025b. Token cleaning: Fine-grained data selection for llm supervised fine-tuning. *arXiv preprint arXiv:2502.01968*.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and 1 others. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. 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)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Leonardo Ranaldi, Giulia Pucci, and André Freitas. 2024. Does the order matter? curriculum learning over languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5212–5220.

Victor Sanh, Albert Webson, Colin Raffel, and et al. 2022. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Zhang Shengyu, Dong Linfeng, Li Xiaoya, Zhang Sen, Sun Xiaofei, Wang Shuhe, Li Jiwei, Runyi Hu, Zhang Tianwei, Fei Wu, and 1 others. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Yi Tay, Shuohang Wang, Anh Tuan Luu, 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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931, Florence, Italy. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. *Preprint*, arXiv:2212.07677.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. *Preprint*, arXiv:2411.12372.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, and et al. 2022. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. *Preprint*, arXiv:1802.03796.

Yong Xie, Karan Aggarwal, and Aitzaz Ahmad. 2024. Efficient continual pre-training for building domain specific large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10184–10201.

Haoran Xu, Seth Ebner, Mahsa Yarmohammadi, Aaron Steven White, Benjamin Van Durme, and Kenton Murray. 2021. Gradual fine-tuning for low-resource domain adaptation. *arXiv preprint arXiv:2103.02205*.

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*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Lu Qianchun. 2025a. Learning like humans: Advancing llm reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6630–6644.

Qi Zhang, Zhiqing Xiao, Ruixuan Xiao, Lirong Gao, and Junbo Zhao. 2025b. D.va: Validate your demonstration first before you use it. *Preprint*, arXiv:2502.13646.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024. Pretraining data detection for large language models: A divergence-based calibration method. *arXiv preprint arXiv:2409.14781*.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, and et al. 2023. LIMA: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Experimental Details

Due to space limitations, we have only provided a brief introduction to our experiment in the main text. In this section, we will introduce the specific experimental details.

Main CPT Settings For the CPT experiments, our fine-tuning dataset consists of 80% code data from CodeParrot and 20% general-domain data from RedPajama, as shown in the table 6. For the calculation of D_{pd}, since we do not have access to the pre-trained dataset, we approximate N_p using the dataset RedPajama. final CPT dataset contains 2 billion tokens. We optimize our models using the AdamW (Loshchilov and Hutter, 2019) optimizer with constant learning rate decay. We set the learning rate to 3e-5 and used a batch size of 64. Training was conducted for 2 epochs with bf16 precision on 8 Ascend 910Bs.

Dataset	Type	Ratio	# Tokens
CodeParrot	Code	80.2%	1.63B
RedPajama	General	19.8%	0.40B
Total	-	100%	2.03B

Table 6: Consist of CPT dataset.

Main SFT Settings For the SFT experiments, we only modified the learning rate and batch size, setting the learning rate to 4e-5 and increasing the batch size to 128. All other settings remain the same as in the CPT experiments.

Two Phase Ablation Settings For ablation study in Section 5.2, in the Only CL setting, both training epochs adopt the structure of Phase 1, where samples are presented in a sorted order based on their difficulty scores. The setting where only Phase 2 is applied is equivalent to the Random baseline, where training data is shuffled without any ordering.

Relationship Between Two Difficulty Metric Settings For the scatter plots shown in Figure 3, we use LLaMA-3.2-1B as the reference model to compute two difficulty metrics, D_{ce} and D_{pd}. We then randomly sample 500 examples from each dataset and plot their respective difficulty values to create scatter plots, alongside reporting the corresponding Pearson correlation coefficients.

Out of Distribution Settings For the out-of-distribution (OOD) evaluation, we assess the model trained on the mathematics dataset with the code benchmark, and the model trained on the code dataset with the mathematics benchmark. All other training hyper-parameters remain identical to those used in the SFT-stage experiments.

B Out of Distribution Experiment Results

Section 5.4 briefly presented OOD results for the four models; the complete figures are shown in Figure 7. As observed, all models suffer some performance degradation on OOD data. **FOT** achieves the best scores on models LLama-3.2-1B and Gemma-3-4B, whereas IT2ACL performs best on models Llama-3.1-8B and Qwen-3-4B. Nonetheless, our approach ranks second on models Llama-3.1-8B and Qwen-3-4B and delivers the highest overall performance, demonstrating that our method more effectively mitigates catastrophic forgetting.

C Additional Results on In-Context Learning

In Section 6, we present the results of the ICL experiments on Llama-3.2-1B; the results on Gemma-3-4B are shown in Table 9. These results likewise support the conclusions drawn in Section 6.

Model	Method	Math		Code				Avg.	
		GSM8K	MATH	HEval	HEval+	MBPP	MBPP+		
Llama-3.2-1B	Random	13.50	3.04	13.41	9.76	14.55	10.32	10.76	0.00
	Anti-Curri	11.52	2.88	16.46	**14.02**	14.29	9.79	11.49	↑0.73
	MIX	12.81	3.12	**16.46**	13.41	15.34	10.85	12.00	↑1.24
	IT2ACL	13.04	**3.56**	12.20	8.54	14.55	10.32	10.37	↓0.39
	Kim and Lee (2024)	12.43	3.00	14.02	10.98	13.76	11.11	10.88	↑0.12
	FOT	**14.86**	3.32	15.24	12.20	**15.34**	**11.38**	**12.06**	**↑1.30**
Llama-3.1-8B	Random	40.94	17.76	54.26	49.39	37.30	32.28	38.66	0.00
	Anti-Curri	38.29	16.66	54.88	49.39	36.77	31.48	37.91	↓0.75
	MIX	40.41	17.84	56.10	**51.83**	37.83	30.95	39.16	↑0.50
	IT2ACL	**42.15**	**18.12**	**56.71**	51.22	37.30	**32.80**	**39.72**	**↑1.06**
	Kim and Lee (2024)	40.56	17.22	53.04	50.00	35.45	30.42	37.57	↓1.09
	FOT	41.70	17.14	55.49	50.00	**39.15**	32.54	39.33	↑0.67
Gemma-3-4B	Random	44.57	19.72	48.78	43.29	43.92	37.30	39.60	0.00
	Anti-Curri	40.94	17.90	45.12	40.24	42.59	36.51	37.22	↓2.38
	MIX	43.29	19.02	46.95	42.68	44.18	37.30	38.90	↓0.70
	IT2ACL	44.20	19.64	48.78	43.90	43.12	36.24	39.31	↓0.29
	Kim and Lee (2024)	45.41	**20.28**	46.95	41.46	45.50	38.36	39.66	↑0.06
	FOT	**45.62**	20.18	**51.22**	**45.12**	**46.03**	**38.89**	**41.18**	**↑1.58**
Qwen-3-4B	Random	51.18	23.54	71.95	65.24	66.14	52.65	55.12	0.00
	Anti-Curri	50.49	23.36	72.56	65.24	64.81	53.44	54.98	↓0.14
	MIX	51.63	23.66	62.80	53.66	54.76	51.59	49.68	↓5.44
	IT2ACL	52.46	24.44	**73.78**	**67.68**	**67.20**	54.50	**56.68**	**↑1.56**
	Kim and Lee (2024)	**52.92**	24.50	71.34	63.41	66.67	53.44	55.38	↑0.26
	FOT	52.31	**25.02**	73.17	64.46	66.14	**54.50**	55.93	↑0.81

Table 7: Out-of-Distribution Result of our methods.

Method	Model			
	LLama3-1B	LLama3-8B	Gemma3-4B	Qwen3-8B
Random	10.76	38.66	39.60	55.12
Anti-Curri	11.49	37.91	37.22	54.98
MIX	12.00	39.16	38.90	49.68
IT2ACL	10.37	**39.72**	39.31	**56.68**
Kim and Lee	10.88	37.57	39.66	55.38
FOT	**12.06**	39.33	**41.18**	55.93

Table 8: Average out-of-distribution results.

Dataset	Difficulty	Demonstration Type		
		None	Easy	Hard
Magicoder	Easy	0.844	0.792 ↓6.2%	0.831 ↓1.5%
	Hard	1.680	1.601 ↓4.6%	1.649 ↓1.8%

Table 9: ICL results on Gemma3-4B.