YuLan-Mini: Pushing the Limits of Open Data-efficient Language Model

Yiwen Hu[1*], **Huatong Song**[1*], **Jie Chen**[1], **Jia Deng**[1], **Jiapeng Wang**[1],
Kun Zhou[2], **Yutao Zhu**[1], **Jinhao Jiang**[1], **Zican Dong**[1], **Yang Lu**[3], **Xu Miao**[3],
Wayne Xin Zhao[1†] and **Ji-Rong Wen**[1]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]University of California, San Diego [3]DataCanvas Alaya NeW
huyiwenwen@foxmail.com,songhuatong123@ruc.edu.cn,batmanfly@gmail.com

Abstract

Due to the immense resource demands and the involved complex techniques, it is still challenging for successfully pre-training a large language models (LLMs) with state-of-the-art performance. In this paper, we explore the key bottlenecks and designs during pre-training, and make the following contributions: (1) a comprehensive investigation into the factors contributing to training instability; (2) a robust optimization approach designed to mitigate training instability effectively; (3) an elaborate data pipeline that integrates data synthesis, data curriculum, and data selection. By integrating the above techniques, we create a rather low-cost training recipe and use it to pre-train YuLan-Mini, a fully-open base model with 2.4B parameters on 1.08T tokens. Remarkably, YuLan-Mini achieves top-tier performance among models of similar parameter scale, with comparable performance to industry-leading models that require significantly more data. To facilitated reproduction, we release the full details of training recipe and data composition. Project details can be accessed at the following link: https://github.com/RUC-GSAI/YuLan-Mini.

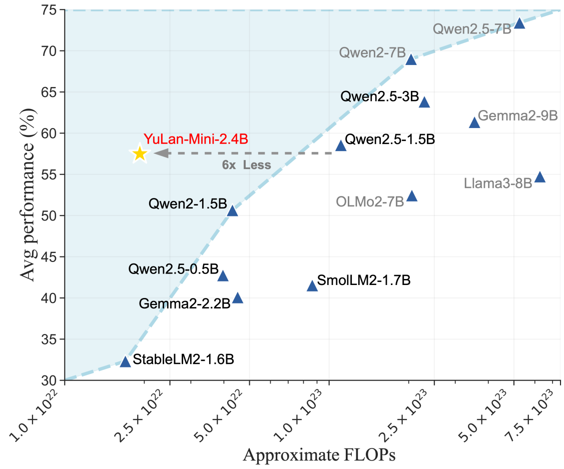


Figure 1: YuLan-Mini achieves performance comparable to Qwen2.5-1.5B on comprehensive benchmarks *i.e.,* MMLU, ARC-Challenge, HellaSwag, WinoGrande, GSM8K, MATH-500, HumanEval, MBPP, and CEval, using only $1/6$ of FLOPs budget, where FLOPs $\approx$ $6 \times$ training tokens $\times$ model size (Kaplan et al., 2020).

1 Introduction

In recent years, large language models (LLMs) have significantly advanced the frontier of AI technology (OpenAI, 2023; Dubey et al., 2024; Bi et al., 2024). It is widely recognized that pre-training is crucial for building the foundational capabilities of the LLMs (Zhao et al., 2023). Although the prevailing pre-training approach of next-token prediction is straightforward, it involves several complexities: First, researchers must design an effective and efficient data pipeline, which typically involves data filtering, data mixing, and data curriculum, as "data" is the most crucial element in enhancing model capabilities. Second, since LLMs consist of a vast number of meticulously organized parameters, accelerating and stabilizing the training process presents a significant challenge. Despite the availability of extensive model checkpoints released by industry companies (Qwen-Team, 2024; Yang et al., 2024b), the core technical details often remain undisclosed in public reports.

Fortunately, the research community has made significant efforts to enhance the availability of data resources (Lozhkov et al., 2024a; Li et al., 2024b; Yu et al., 2025) and the openness of pre-training methodologies (Allal et al., 2024; AllenAi, 2024; Zhang et al., 2024a). These contributions offer basic technical approaches and essential resources for pre-training an LLM. Despite these advancements, *open LLMs*—those with fully disclosed technical details still face main limitations of under-performance compared to industry counterparts, or requiring large data and computational resources. Therefore, developing competitive LLMs

*Equal Contribution
†Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5374–5400

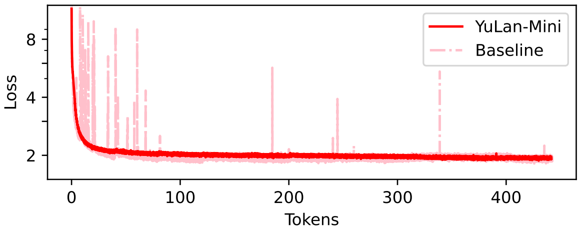


Figure 2: YuLan-Mini improves final training stability under large learning rate and deep architecture.

with limited training resources remains a challenge, particularly in university-level laboratories.

Motivated by the above considerations, our contributions in are as follows: **(1)** We present a *fully-open* 2.4B-parameter language model, YuLan-Mini, that achieves top-tier performance among models of similar parameter scale. To facilitate reproduction, we report the complete training details for YuLan-Mini, including all data composition for training curriculum, training source code, and optimizer states. **(2)** We devise an elaborately designed *efficient data pipeline* that compiles data synthesis, data curriculum, and data selection. In particular, we extensively leverage an classifier-based "easy-to-hard" curriculum learning and synthetic data like formal mathematics reasoning. **(3)** We investigate deeply into the transformers architecture and provide an *efficient pre-training method* that effectively mitigates training instability. We identify several training instability factors *e.g.,* exploding hidden states and RMSNorm representation collapse. By exploring a variety of techniques to stabilize under radical configuration and enhance the performance of YuLan-Mini.

To demonstrate the effectiveness of our efficient pre-training methodologies, we compare it with a few competitive base models from both research and industry on a variety of benchmarks. We also conduct extensive ablation experiments on our training stability methods (Section 6.1) and data pipeline (Section 6.2). Experimental results show that our base model, YuLan-Mini, can achieve very promising results among these compared models (Figure 1). For instance, it outperforms recent models *e.g.,* OLMo2-7B, SmolLM2-1.7B, and Llama3-8B.

2 Related Work

Pre-Training of LLMs The prevailing pre-training approach often incurs significant costs (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023). Therefore, much recent research has focused on optimizing the performance of relatively small language models (Zhang et al., 2024b; Hu et al., 2024; Liu et al., 2024c; Bellagente et al., 2024; Allal et al., 2024). Existing research on Transformer training stability has identified various sources of instability and proposed mitigation methods (Yang et al., 2022; Takase et al., 2023; Nishida et al., 2024). However, few studies have examined training stability from the perspective of efficiency. For instance, while QK LayerNorm improves stability, it adds computational overhead (Henry et al., 2020; Bellagente et al., 2024; Rybakov et al., 2024). The μP method stabilizes early training but still faces instability under large learning rates (Yang et al., 2022). Similarly, reducing the AdamW epsilon parameter works well only for larger models (Molybog et al., 2023; Wortsman et al., 2024), while techniques like Z-Loss and weight decay offer limited benefits (Zoph et al., 2022b).

Pre-Training data pipeline Data pipelines generally involve data filtering, curriculum learning, and data synthesis (Young et al., 2024). Data filtering eliminates redundant data using methods like de-duplication (Sun et al., 2024), model-based scoring (Lozhkov et al., 2024a), or gradient-based selection (Xia et al., 2024). Curriculum learning adjusts the order of data across training stages (Zhu et al., 2024), while data synthesis leverages existing models to integrate posterior insights (*e.g.,* specific topics) (Gunasekar et al., 2023; Wei et al., 2024; Chen et al., 2024). However, most research work focuses on isolated components, and industrial models seldom reveal pipeline details.

3 Efficient Pre-Training

Training instability poses a significant challenge to the effective training of LLMs, *e.g.,* irrecoverable divergent training. While large learning rates or deep architectures can accelerate model convergence, this improvement is only feasible as long as there are no loss spikes or an escalating gradient norm (AllenAi, 2024). Our training approach combines such configuration with improved stability, enabling performance on par with industry-level models while using significantly fewer resources.

3.1 Architecture Improvements

We summarize our architecture details in Table 1. Specifically, YuLan-Mini employs a 2.4B LLaMA-

Table 1: Architecture comparison between LLMs featuring training stability.

Methods	MiniCPM	OLMo2 7B	YuLan-Mini
Arch	Shallow	Shallow	Deep
Param Init	μP	/	μP
Numerical	/	/	Re-Param
LayerNorm	Pre-LN	Reordered	Pre-LN
Residual	Scale	/	Scale
Attention	/	QK-Norm	/
Embedding	Tie+Scale	w/o WD	Tie+Scale
Peak LR	0.01	3×10^{-4}	0.01
Avg Perf	49.5	52.5	57.5

like transformer architecture with embedding tying (Press and Wolf, 2017). The decoder layer can be formalized as:

$$\boldsymbol{z}^l = \boldsymbol{y}^l + \mathtt{FFN}(\mathtt{RMSNorm}(\boldsymbol{y}^l)), \tag{1}$$

$$\boldsymbol{y}^l = \boldsymbol{x}^l + \mathtt{MHA}(\mathtt{RMSNorm}(\boldsymbol{x}^l)), \tag{2}$$

where $\boldsymbol{x}^l, \boldsymbol{y}^l, \boldsymbol{z}^l$ are hidden states of each layer l and $\boldsymbol{u} = \mathtt{RMSNorm}(\boldsymbol{x}^l)$ and $\boldsymbol{v} = \mathtt{RMSNorm}(\boldsymbol{y}^l)$ are $\mathtt{RMSNorm}$ outputs. For training efficiency, we specifically use large global learning rate of 0.01 and a deep and thin architecture (56 decoder layers). We combine a parameter initialization approach akin to μP with matrix-level re-parameterization to stabilize training under this configuration. We estimate a calculation-efficient vocabulary size of 99K, and apply BPE-dropout ($p = 0.2$) (Provilkov et al., 2020) and individual-digit tokenization to further balance the update of embedding. We leverage several fused kernels (Hsu et al., 2024; Dao, 2024) to enhance efficient calculation, achieving a 51.57% Model FLOPs Utilization (MFU). The detailed overall configuration is provided in Appendix A.

3.2 Bounding Dynamics of Transformers to Mitigate Abnormal Gradients

After analyzing the training dynamics of our model, we observe that hidden states (*a.k.a.,* activations) can reveal deeper underlying issues which are difficult to detect in the early stages when focusing solely on the loss (Figure 3 Right). Specifically, hidden states diverge increasingly with model depth (*i.e.,* more layers) and, more significantly, exhibit an exponential upward trend with increasing training steps. This empirically results in substantial gradient updates, which, in turn, can lead to training instability. To address this, we next estimate the bounds of hidden states in transformers,

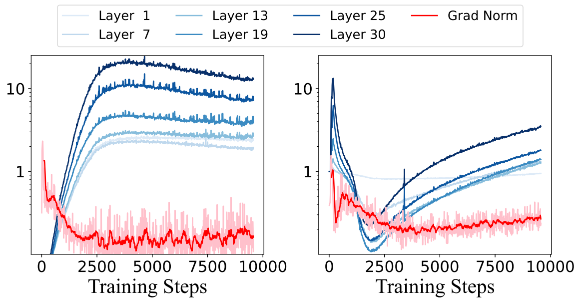


Figure 3: Comparison of training dynamics (hidden state variance and gradient norm) between convergent (left) and divergent (right) trials on a log-scale. Both trials exhibit consistent loss, but the divergent trial shows increasing hidden state variance and gradient norm.

forming the foundation for the development of mitigation strategies in Section 3.3.

Residual connection To investigate the growing hidden states and subsequent exploding gradient across model depth (Figure 3), we analyze the variance addition of each layer $\Delta H^l = \mathrm{var}(\boldsymbol{z}^l) - \mathrm{var}(\boldsymbol{x}^l) = \mathrm{var}(\mathtt{MHA}(\boldsymbol{v})) + \mathrm{var}(\mathtt{FFN}(\boldsymbol{u}))$. By plugging in the variance of $\mathtt{MHA}$ and $\mathtt{FFN}$ into Equation 1 and 2, we can estimate the upper bound of variance addition in initial steps as:

$$\Delta H^l < d_{\text{model}}^2 \cdot \mathrm{var}(\mathbf{W}_v) \cdot \mathrm{var}(\mathbf{W}_o) + d_{\text{ffn}} \cdot d_{\text{model}} \cdot \mathrm{var}(\mathbf{W}_{up}) \cdot \mathrm{var}(\mathbf{W}_{down}), \tag{3}$$

which greatly accumulates across decoder layers. A detailed derivation can be found in Appendix C and Takase et al. (2023).

Layer normalization RMSNorm is proposed to re-scale data, providing scale-insensitivity to models (Zhang and Sennrich, 2019). We observe a behavior in Layer Normalization (LN) commonly associated with training instability, referred to as "*RMSNorm representation collapse*". In this phenomenon, the LN outputs rapidly collapse to a very small variance, which can lead to spikes in attention weights and loss (Figure 4). Previous work suggests that the variance of RMSNorm inputs should be ≥ 1, as values below this threshold can lead to gradient inflation (Takase et al., 2023):

$$\left\| \frac{\partial \mathtt{RMSNorm}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 = \mathcal{O}\left(\frac{\sqrt{d}}{\|\boldsymbol{x}\|_2} \right),$$

which suggests initializing embeddings to 1 or employing more complex techniques, such as separate weight decay on embedding (AllenAi, 2024) or

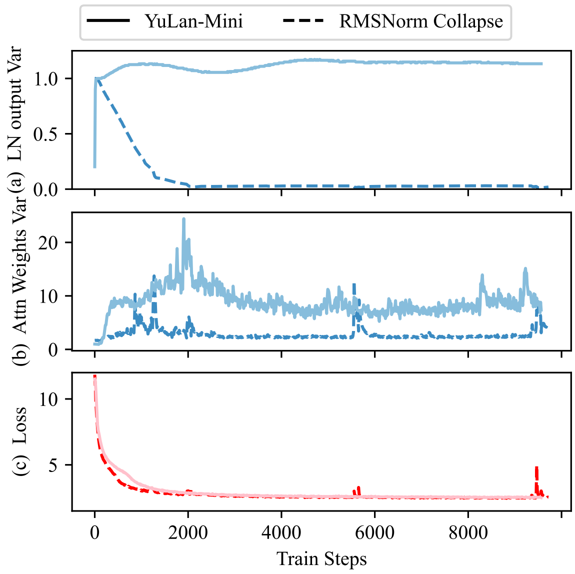


Figure 4: RMSNorm representation collapse. The output of LN collapsing to small values may lead to instability.

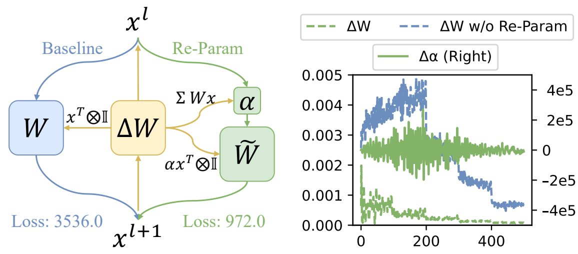
embedding normalization (Scao et al., 2022). However, we find these methods can trigger RMSNorm representation collapse, by hindering necessary updates of the scale vector $\boldsymbol{g}$ in it.

3.3 Mitigating Instability through μP and Re-parameterization

To mitigate training instability, we employ a two-pronged approach: 1) preventing growing hidden states and RMSNorm representation collapse through carefully designed model initialization, and 2) absorbing large gradient variability via matrix re-parameterization:

Consistent architecture Compared to original scaled initialization (Shoeybi et al., 2020; Takase et al., 2023), the Maximal Update Parametrization (μP) has been proposed (Yang et al., 2022, 2024c) to provide a consistent architecture for model initialization and scaling, including embedding scaler, residual scaler, learning rate scaler, and scaled initialization. μP mitigate training instability within transformers architecture. For instance, the scaled initialization initialize `MHA` and `FFN` with small values $\text{std}(\mathbf{W}_v) = \text{std}(\mathbf{W}_{up}) = \sqrt{2/(5d_{model})}$ and $\text{std}(\mathbf{W}_o) = \text{std}(\mathbf{W}_{down}) = \sqrt{1/(5d_{model} \cdot n_{layers})}$, thereby mitigating growing hidden states rooted across all hidden layers shown in Equation 3:

$$\sum_{l=1}^{n_{layers}} \Delta H^l < \frac{7}{25}.$$



(a) Derivatives of Re-Param. (b) Gradient norm.

Figure 5: Re-Param enhances gradient representation by "absorbing" large gradient variability to $\Delta\alpha$.

Besides μP, we also incorporate embedding tying by initializing the embeddings with a variance smaller than 1. This helps prevent RMSNorm representation collapse by enabling updates to RMSNorm during the early stages of training.

Gradient representation However, we observe that spikes in loss still occur with large learning rates when using μP. We empirically find that this is suffered from variability in gradient updates. Inspired by recent studies in training instability (Nishida et al., 2024; Chung et al., 2024), we find re-parameterization (Re-Param) method provides a different gradient representation as illustrated in Figure 5a:

$$\mathbf{W} = \alpha\widetilde{\mathbf{W}}, \alpha \in \mathbb{R},$$

where the matrix weights $\mathbf{W}$ is re-parameterized with an additional learnable parameter α. Our surrogate experiments on a simple linear regression show that Re-Param successfully decompose the original gradient and absorb the variability of it. Combined with the consistent architecture provided by μP, we find the above Re-Param method to be effective in addressing exploding hidden states and thereby enhance pre-training efficiency.

4 Efficient Data Pipeline

Effective data curation and curriculum design have been shown to be key to improving model performance when the data volume for training is fixed. However, few open studies provide full technical details about the entire data pipeline. In this section, we present a comprehensive, efficient, and fully open data pipeline that includes filtering data, synthesizing high-quality reasoning data, optimizing training data scheduling, and improving data selection during the annealing stage. By utilizing only 1.08T of training data, we achieve industry-level

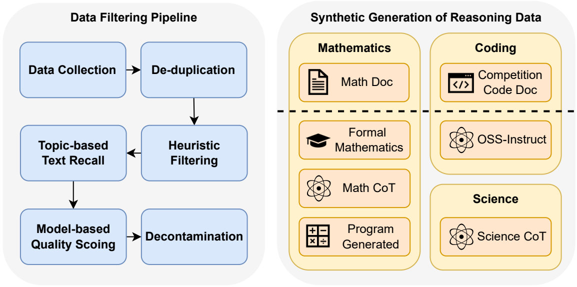


Figure 6: Illustration of our data filtering and synthetic for reasoning data pipeline.

results with relatively low cost. Figure 6 illustrates the data filtering and synthesis process, with the implementation details provided in Appendix D.

4.1 Synthetic Generation of Reasoning Data

Reasoning is a crucial skill for LLMs (Huang and Chang, 2023), but real-world datasets often lack texts with complex reasoning. Recent research indicates that reasoning structures are important to enhance a model's reasoning abilities (Yang et al., 2025; Li et al., 2025). In YuLan-Mini, we propose an efficient approach to systematically scale reasoning structures, leading to significant improvements in mathematical and coding capabilities. We show in Appendix E that this does not compromise the subsequent post-training capability.

Formal theorem proving Lean provides a verifiable environment to explore theorem proving formally, which has been shown effective in improving mathematical reasoning (Xin et al., 2024; Ying et al., 2024b). As far as we know, we are the first public study to introduce formal mathematics data in pre-training, using a total amount of 0.2B lean-based synthesized data.

Reasoning primitives In addition to the "*predict the next tactic*" used in existing formal theorem proving research (Ying et al., 2024a; Wu et al., 2024), we extend it to three new reasoning primitives: (1) Deduction: $\text{State}_{\text{before}}, \text{Tactic} \rightarrow \text{State}_{\text{after}}$; (2) Abduction: $\text{State}_{\text{after}}, \text{Tactic} \rightarrow \text{State}_{\text{before}}$; and (3) Induction: $\text{State}_{\text{before}}, \text{State}_{\text{after}} \rightarrow \text{Tactic}$.

CoT reasoning We generate CoT reasoning data for three fields: mathematics, coding, and science, by using instruct version of `Qwen2.5-7B` and `Qwen2.5-Math-7B`. Additionally, we develop a program to automatically convert simple mathematical queries (*e.g.,* "`What is 0.079 + 162?`") into detailed calculation procedures.

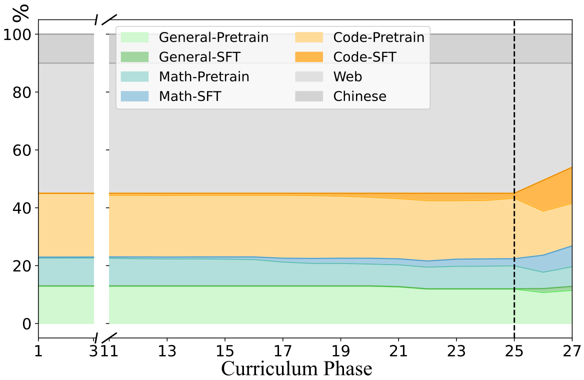


Figure 7: The data mixture proportion. The annealing stage begin after the dashed line.

Reflection To enhance model's reasoning ability, we incorporate the reflection mechanism for solving math problems. We use `Qwen2.5-7B-Instruct` to generate both correct and incorrect solutions with corresponding error analysis to form a synthetic reflection process. This enhances model's reasoning ability without reinforcement learning.

4.2 Data Curriculum

Data curriculum intuitively aligns with the learning process of LLMs, but existing research rarely achieves real-world effectiveness due to its large costs. Our approach offers a potential solution for small corpus (*e.g.,* 1T tokens). Building on the WSD three-stage learning rate scheduler, we further divide the process into 27 stages, each spanning 40B tokens. We dynamically design the curriculum based on content difficulty and model capability while keeping adjustments within 3% to avoid loss spikes. We primarily implement curriculum learning in mathematics and coding content. Figure 7 illustrates the data distribution for each curriculum phase.

Content difficulty Text of varying difficulty levels are unevenly distributed in datasets. Typically, we reorder and perform weighted sampling on the content according to difficulty, which facilitates an efficient learning process. To estimate a difficulty level, we primarily use quality classifiers such as `fineweb-edu-scorer` and `python-edu-scorer`. We heuristically analyze the difficulty distribution across score segments to ensure the curriculum is correctly ordered due to its inherent bias.[1]

[1] For instance, when using the python-edu-scorer, low scores in large datasets often correspond to noisy data, whereas in meticulously curated datasets, low scores typically repre-

Dynamic model capabilities For each curriculum phase, we reassess the model's overall performance and adjust the data ratio based on it. For example, if the model presents strong performance in HumanEval, we may consider decrease the amount of code data in subsequent phases. To further improve its reasoning ability, a small amount (<5%) of instruction data is introduced to the later stage of stable stage, and is increased to 19.19% in the annealing stage. Specifically, we incorporate the formal mathematical reasoning data (theorem proving in Lean) and advanced reasoning data (Section 4.1).

4.3 Data Selection for Annealing Stage

Selecting high-quality data during the annealing stage is crucial, as learning rate annealing enables the model to rapidly improve its performance (Hu et al., 2024). For this reason, we carefully curate high-quality data for the annealing process. Previous studies on data selection often yield suboptimal results or incur significant computational overhead (Xia et al., 2024). Thus, we mainly consider an improved LESS method (Xia et al., 2024), combining the method InsTag (Lu et al., 2024) for constructing a diversified target set (a subset of training set). Specifically, we replace the random matrix used in the gradient mapping with a matrix derived through PCA dimensionality reduction on the target set. Furthermore, we observe that the gradients at each layer are nearly orthogonal, allowing us to remove certain layers to enhance efficiency.

5 Experiments

Experimental results of different base models on public benchmarks are shown in Table 2, and we can make the following observations:

• *Superior training efficacy.* Overall, YuLan-Mini achieves highly competitive performance compared to leading small industry models, despite being trained on just 1.08T tokens. Meanwhile, most of our training data comes from open-source and synthetic datasets, demonstrating that with careful data cleaning, selection, and scheduling, we develop a robust base model even with limited resources in a university-level laboratory.

• *Excellence in mathematical and coding.* On specific benchmarks for mathematical reasoning (MATH-500 and GSM8K) and coding generation (HumanEval and MBPP), YuLan-Mini achieves leading performance. This consistent superiority can be mainly attributed to the use of high-quality pre-training corpus and reasoning synthetic data (*e.g.,* formal mathematics reasoning problems). Our core idea is to extend the types of reasoning data and enhance the complex reasoning capacities of our base model, which leads to large improvements on mathematical benchmarks.

• *Strong general capability.* Beyond specialized tasks, YuLan-Mini also demonstrates strong performance on various general benchmarks, spanning from language modeling and commonsense reasoning, highlighting the versatility of the model. It indicates that our pre-training approach well balances the learning of diverse abilities, resulting in a robust general-purpose foundation model.

Details of the benchmarks and evaluation settings are provided in Appendix B.

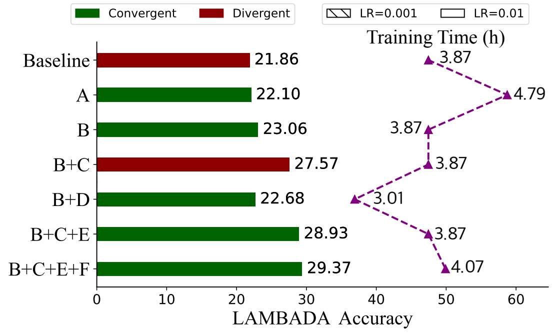
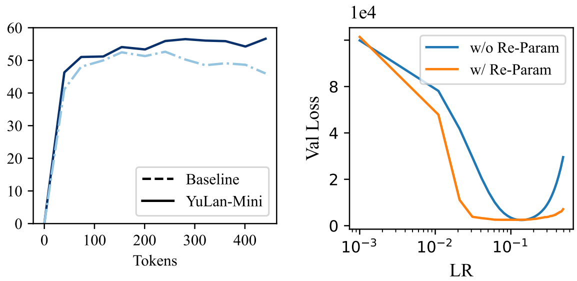


Figure 8: Ablation experiments on training instability mitigation methods: (A) QK LayerNorm, (B) Weight Decay, (C) Cerebrase μP, (D) Shallow and Wide Architecture, (E) Depth μP, (F) Re-Param.



(a) Our stable training recipe improves model capability. (b) Re-Param offers insensitivity to learning rate.

Figure 9: Ablation study of overall training recipe and Re-Param.

6 Ablation Study

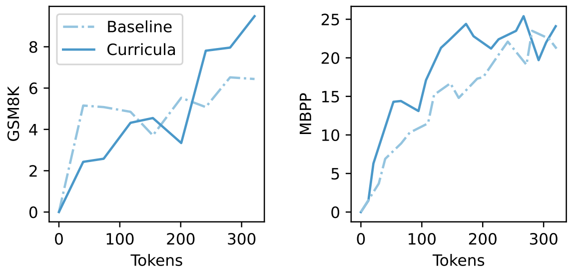
6.1 Methods of Mitigating Training Instability

Surrogate experiments on Re-Param We conduct surrogate linear regression experiments to

sent high-quality competition-level problems.

Table 2: Performance on math, code, and reasoning benchmarks. Results marked with * are cited from their official paper or report. The best and second best results (±1.0) are **bold** and underlined, respectively.

Models	Model Size	Data Size	MATH 500	GSM 8K	Human Eval	MBPP	MMLU	CEval	ARC-c	Hella Swag	Wino Grande	Avg
MiniCPM	2.7B	1T	15.0	53.8	50.0*	47.3	53.4	48.2	43.9	67.9	65.7	49.5
Qwen2	1.5B	7T	22.6	46.9*	34.8*	46.9*	55.9	**71.9**	42.9	66.1	66.1	50.5
Qwen2.5	0.5B	18T	23.6	41.6*	30.5*	39.3*	47.5	54.3	39.5	50.5	55.9	42.5
Qwen2.5	1.5B	18T	**45.4**	**68.5***	37.2*	60.7	60.2*	69.1	53.4	67.2	64.5	**58.5**
Gemma2	2.6B	2T	18.3*	30.3*	19.5*	42.1*	52.2*	28.0*	**55.7***	**74.6***	**71.5***	43.6
StableLM2	1.6B	2T	1.8	20.6	8.5	17.5	40.4	27.0	40.8	69.8	64.6	32.3
SmolLM2	1.7B	11T	11.8	31.1*	23.4	45.0	51.9	35.1	35.5	73.0	67.4	41.6
Llama3.2	3.2B	/	7.4	3.2	29.3	49.7	**63.4**	44.4	48.8	**75.6**	67.5	43.3
Falcon3	3.2B	/	**44.6**	66.0	34.4	52.5	59.7	38.2	51.6	65.8	64.4	53.0
YuLan-Mini	2.4B	1T	37.8	**68.5**	**64.0**	**65.9**	49.1	48.2	49.3	67.2	67.2	**57.5**



(a) Math. (b) Code.

Figure 10: Performance of different data curricula on math and code benchmarks.

validate the effectiveness of Re-Param, as discussed in Section 3.3. Specifically, we train a 20,000-dimensional linear regression model using the Adam optimizer. Our results demonstrate that Re-Param improves insensitivity to the learning rate by decomposing gradient variability into a learnable factor. This method effectively stabilizes training across a wide range of learning rates.

Main training recipe The effectiveness of our pre-training recipe mainly comes from a combination use of μP and re-parameterization, which also provides: (1) consistent training dynamics, including training loss, gradient norm, and hidden states, (2) enhanced model capabilities in language modeling and generation (Figure 9a Left), and (3) stable model weights (Figure 9a Right).

We provide ablation study on our recipe in Figure 8. Unlike previous studies that focus on test loss (AllenAi, 2024), our work primarily examines LAMBADA accuracy, which we observe can behave differently despite comparable test loss. We build a 0.2B proxy model with a deep and thin architecture resembling YuLan-Mini and train it on 20B tokens. The main observations are as follows.

- *QK LayerNorm*. This method addresses gradient divergence (green bar) but introduces a 24% runtime overhead. However, it has a similar loss, with no additional improvement in LAMBADA.
- *Weight decay*. Using weight decay achieves comparable stabilization and 23.06% accuracy without computational penalty.
- *Cerebrase μP*. Combining Cerebrase μP with larger learning rate yields improvements, but loss spikes occur and ultimately lead to divergence.
- *Shallow architecture*. Shallow and wide model are less likely encounter training instability even in large LR, but fails to deliver better performance.
- *Depth μP*. By scaling down `FFN` and `MHA` in residual, Depth μP provides further stabilization besides Cerebras μP in our deep architecture.
- *Re-Param*. Our solution achieves peak performance (29.37% accuracy) through absorbing variability in large gradients, while introducing only 5% additional runtime compared to baseline.

6.2 Ablation Study on Data Pipeline

Synthetic data We utilize various data synthesis methods, as outlined in Section 4.1. The key observations regarding the use of formal mathematics data (*i.e.,* Lean theorem proving) during the learning rate annealing stage are as follows: (1) *w. Lean* incorporates 0.1B Lean data into the annealing data (80B tokens), and (2) *w/o Lean* incorporates 0.1B web data into the annealing data. As shown in Table 4, the integration of formal mathematical data notably enhances the model's mathematical capabilities, even when incorporating non-formal math. This results in a 2.7% improvement on GSM8K and a 16.4% improvement on MATH-500, with the most significant gains observed on the more challenging problems (*i.e.,* MATH-500). Importantly, the inclusion of Lean data does not affect the model's generative capabilities.

Table 3: Performance on math, code and reasoning benchmarks. The best result is **bold**.

Models	MATH	GSM8K	HumanEval	MBPP	MMLU	CEval	ARC-c	GPQA	IFeval	Avg
Qwen-2.5-1.5B-Instruct	**55.2**	73.2	61.6	**88.1**	57.5	**65.4**	47.8	29.8	42.5	57.9
Llama3.2-3B-Instruct	48.0	43.4	51.5	80.4	**60.0**	45.9	**78.6**	**38.6**	-	55.8
YuLan-Mini-Instruct	**55.2**	**81.8**	**67.7**	85.7	53.6	50.5	51.8	30.1	**44.0**	57.8

Table 4: Performance on math benchmarks during the annealing stage with and without Lean data.

Setting	GSM8K	MATH-500	LAMBADA
(1) *w/o Lean*	66.65	32.6	64.72
(2) *w. Lean*	68.46	39	65.67

Table 5: Ablation study on our data selection method. **HE** refers to HumanEval.

Method	LAMBADA	MMLU	GSM8K	HE	Time
(1) *Random*	54.6	38.3	31.6	36.8	-
(2) *LESS*	50.9	38.6	31.5	33.1	3.5h
(3) *w. PCA*	52.6	41.4	30.4	30.0	3.5h
(4) *w. LR*	51.7	37.8	35.1	36.7	1h
(5) *Ours*	56.4	40.9	40.3	34.9	1h

Curriculum learning We choose GSM8K and MBPP benchmarks to measure the effectiveness of our data curriculum. As shown in Figure 10, a gradually increasing difficulty level (math and code curricula) is more beneficial compared to a reversed "hard-to-easy" curriculum (math baseline), or a randomly shuffled difficulty order (code baseline). Specifically, on the GSM8K dataset, the math baseline's "hard-to-easy" approach leads to faster initial performance gains. However, as high-difficulty content is quickly exhausted, the "easy-to-hard" strategy surpasses it in the later stages (Figure 10a). On the MBPP dataset, according to our investigation, when simpler data is used in the early stages of training, the model can quickly master basic coding skills (such as basic operations with lists and dictionaries). As training progresses, model can gradually learn more advanced coding operations (Figure 10b).

Data selection for micro-annealing Here we validate the effectiveness of our data selection method employed during the annealing phase through micro-annealing surrogate experiments (Section 4.3). We examine five distinct configurations: (1) *Random* selects data randomly; (2) *LESS* represents the original LESS method for data selecting; (3) *LESS w. PCA* uses the PCA matrix obtained from the target set for projection; (4) *LESS w. LR* removes 80% of the layers from the original model; (5) *LESS w. PCA & LR* is our enhanced LESS method. We perform data selection on 0.16B instructional tokens, retaining the top 50% based on scores, and utilize a 0.42B pre-training dataset to maintain the data distribution. As shown in Table 5, substituting a random matrix with a PCA matrix for projection generally enhances model performance. Notably, removing 80% of the layers can increase selection speed by 3.5 times, and enhance the performance.

6.3 Post-training Performance

We conduct post-training for YuLan-Mini. We first fine-tune YuLan-Mini on collected high-quality datasets, then utilize the DPO and PPO algorithm to further fine-tune our model on human alignment and complex reasoning datasets. The experiment details can be found in Appendix E. As the results shown in Table 3, we can see our YuLan-mini also exhibits better performance than these competitive baselines, indicating its learned strong capability from our designed pre-training method.

7 Conclusion

In this paper, we introduced YuLan-Mini, a highly capable base model comprising 2.42 billion parameters. We provided comprehensive technical details and resources, including the composition of the training curriculum, the source code, and the optimizer state. We investigated the causes of training instability and proposed an effective method for stabilizing the training process. Furthermore, we designed a complete and efficient data pipeline, detailing the synthesis of high-quality reasoning data, the design of the data curriculum, and the selection of data during the annealing phase. The advanced stabilization techniques and meticulously organized data pipeline enabled us to conduct efficient pre-training, achieving commendable performance with only 1.08T tokens.

Limitations

In this paper, we explore the training stability of large language models during pre-training and present a comprehensive data pipeline. Utilizing only 1.08T tokens, we successfully trained a highly effective base model with 2.4 billion parameters, demonstrating the efficiency of our training approach. But there are also two limitations in this work. Firstly, due to the substantial computational resources required for pre-training, and given that we operate within a university-level laboratory with constrained computing capabilities. We currently have only 48 A800 GPUs, which limits us to training a smaller model with 2.4 billion parameters. Similarly, due to hardware constraints, we can not explore more efficient pre-training using FP8. Secondly, due to the extensive volume of training data, comprising 1.08 trillion tokens, we only conduct data curriculum ablation experiments on approximately 400 billion tokens and we are unable to perform a comprehensive ablation study on the data curriculum encompassing the entirety of the training process.

Ethics Statement

We abide by ethical norms. We adhere to the relevant licenses and usage guidelines for the datasets, ensuring that no personal or offensive information is included. Documentation for the datasets is available in our project repository. We only use the AI assistant during the paper refinement process.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grant No. 92470205 and 62222215, Beijing Municipal Science and Technology Project under Grant No. Z231100010323009 and Beijing Natural Science Foundation under Grant No. L233008. Xin Zhao is the corresponding author.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Lewis Tunstall, Agustín Piqueres, Andres Marafioti, Cyril Zakka, Leandro von Werra, and Thomas Wolf. 2024. Smollm2 - with great data, comes great performance.

AllenAi. 2024. OLMo 2: The best fully open language model to date. blog post.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, Meng Lee, Emad Mostaque, Michael Pieler, Nikhil Pinnaparaju, Paulo Rocha, Harry Saini, Hannah Teufel, Niccoló Zanichelli, and Carlos Riquelme. 2024. Stable LM 2 1.6b technical report. *CoRR*, abs/2402.17834.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, Alex X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. 2024. Deepseek LLM: scaling open-source language models with longtermism. *CoRR*, abs/2401.02954.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Jie Chen, Zhipeng Chen, Jiapeng Wang, Kun Zhou, Yutao Zhu, Jinhao Jiang, Yingqian Min, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, Yankai Lin, Ruihua Song, Jun Xu, Xu Chen, Rui Yan, Zhewei Wei, Di Hu, Wenbing Huang, and Ji-Rong Wen. 2024. Towards effective and efficient continual pre-training of large language models. *CoRR*, abs/2407.18743.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Woojin Chung, Jiwoo Hong, Na Min An, James Thorne, and Se-Young Yun. 2024. Stable language model pre-training by reducing embedding variability. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 10852–10863. Association for Computational Linguistics.

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

Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Alexandre de Brébisson and Pascal Vincent. 2016. The z-loss: a shift and scale invariant classification loss belonging to the spherical family. *CoRR*, abs/1604.08859.

Nolan Dey, Gurpreet Gosal, Zhiming Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. 2023a. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *CoRR*, abs/2304.03208.

Nolan Dey, Gurpreet Gosal, Zhiming, Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. 2023b. Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster. *arXiv preprint*. ArXiv:2304.03208 [cs].

Hantian Ding, Zijian Wang, Giovanni Paolini, Varun Kumar, Anoop Deoras, Dan Roth, and Stefano Soatto. 2024. Fewer truncations improve language modeling. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Falcon-LLM Team. 2024. The falcon 3 family of open models.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2024. How to train long-context language models (effectively). *CoRR*, abs/2410.02660.

Gemma Team. 2024. Gemma.

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

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *CoRR*, abs/2401.14196.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. 2020. Query-Key Normalization for Transformers. *arXiv preprint*. ArXiv:2010.04245 [cs].

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. 2024. Liger kernel: Efficient triton kernels for LLM training. *CoRR*, abs/2410.10989.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zhen Leng Thai, Kai Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *CoRR*, abs/2404.06395.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1049–1065. Association for Computational Linguistics.

Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. 2024. OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models. *CoRR*, abs/2411.04905.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv preprint*. Issue: arXiv:1909.10351 1097 citations (Semantic Scholar/arXiv) [2023-07-31] arXiv:1909.10351 [cs].

J. Kaplan, Sam McCandlish, T. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *ArXiv*.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. LAMBADA: backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pages 6547–6568. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *Preprint*, arXiv:1704.04683.

Joonhyung Lee, Jeongin Bae, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. 2024. To FP8 and back again: Quantifying the effects of reducing precision on LLM training stability. *CoRR*, abs/2405.18710.

Conglong Li, Minjia Zhang, and Yuxiong He. 2022. The stability-efficiency dilemma: Investigating sequence length warmup for training GPT models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Shishir G. Patil, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. LLMs Can Easily Learn to Reason from Demonstrations Structure, not content, is what matters! (arXiv:2502.07374).

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024a. CMMLU: measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11260–11285. Association for Computational Linguistics.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M. Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Raghavi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. 2024b. Datacomp-lm: In search of the next generation of training sets for language models. *CoRR*, abs/2406.11794.

Xinyu Lian, Sam Ade Jacobs, Lev Kurilenko, Masahiro Tanaka, Stas Bekman, Olatunji Ruwase, and Minjia Zhang. 2024. Universal checkpointing: Efficient and flexible checkpointing for large scale distributed training. *CoRR*, abs/2406.18820.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. 2024a. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*.

Xiaoran Liu, Kai Lv, Qipeng Guo, Hang Yan, Conghui He, Xipeng Qiu, and Dahua Lin. 2024b. Longwanjuan: Towards systematic measurement for long text quality. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 5709–5725. Association for Computational Linguistics.

Zechun Liu, Changsheng Zhao, Forrest N. Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024c. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024a. Fineweb-edu.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian J. McAuley, Han Hu, Torsten Scholak, Sébastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, and et al. 2024b. Starcoder 2 and the stack v2: The next generation. *CoRR*, abs/2402.19173.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. 2023. A theory on adam instability in large-scale machine learning. *CoRR*, abs/2304.09871.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *Preprint*, arXiv:1604.01696.

Kosuke Nishida, Kyosuke Nishida, and Kuniko Saito. 2024. Initialization of large language models via reparameterization to mitigate loss spikes. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 22699–22714. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2024. Openwebmath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. *arXiv preprint*. ArXiv:2406.17557.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 157–163. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. Bpe-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1882–1892. Association for Computational Linguistics.

Qwen-Team. 2024. Qwen2.5: A party of foundation models.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Oleg Rybakov, Mike Chrzanowski, Peter Dykas, Jinze Xue, and Ben Lanir. 2024. Methods of improving LLM training stability. *CoRR*, abs/2410.16682.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M. Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. 2022. What Language Model to Train if You Have One Million GPU Hours? *arXiv preprint*. ArXiv:2210.15424 [cs].

Noam Shazeer. 2020. GLU variants improve transformer. *CoRR*, abs/2002.05202.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism. *Preprint*, arXiv:1909.08053.

Yiding Sun, Feng Wang, Yutao Zhu, Wayne Xin Zhao, and Jiaxin Mao. 2024. An integrated data processing framework for pretraining foundation models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2713–2718. ACM.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pre-training of large language models. *CoRR*, abs/2312.16903.

Tianyi Tang, Hu Yiwen, Bingqian Li, Wenyang Luo, ZiJing Qin, Haoxiang Sun, Jiapeng Wang, Shiyi Xu, Xiaoxue Cheng, Geyang Guo, Han Peng, Bowen Zheng, Yiru Tang, Yingqian Min, Yushuo Chen, Jie Chen, Ranchi Zhao, Luran Ding, Yuhao Wang, Zican Dong, Xia Chunxuan, Junyi Li, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2024. LLMBox: A Comprehensive Library for Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 388–399, Bangkok, Thailand. Association for Computational Linguistics.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2023. D4: improving LLM pretraining via document de-duplication and diversification. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Howe Tissue, Venus Wang, and Lu Wang. 2024. Scaling law with learning rate annealing. *CoRR*, abs/2408.11029.

Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *CoRR*, abs/2405.17067.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie E. Everett, Alexander A. Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. 2024. Small-scale proxies for large-scale transformer training instabilities. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zijian Wu, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean-github: Compiling github LEAN repositories for a versatile LEAN prover. *CoRR*, abs/2407.17227.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *CoRR*, abs/2405.14333.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2024. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4643–4663. Association for Computational Linguistics.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2578–2589. Association for Computational Linguistics.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin

Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *CoRR*, abs/2309.10305.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *CoRR*, abs/2407.10671.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2022. Tensor programs V: tuning large neural networks via zero-shot hyperparameter transfer. *CoRR*, abs/2203.03466.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. 2024c. Tensor programs VI: feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. ReasonFlux: Hierarchical LLM Reasoning via Scaling Thought Templates. (arXiv:2502.06772).

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024a. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *CoRR*, abs/2406.03847.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. 2024b. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.

Andy B. Yoo, Morris A. Jette, and Mark Grondona. 2003. SLURM: simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing, 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003, Revised Papers*, volume 2862 of *Lecture Notes in Computer Science*, pages 44–60. Springer.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Yijiong Yu, Ziyun Dai, Zekun Wang, Wei Wang, Ran Chen, and Ji Pei. 2025. Opencsg chinese corpus: A series of high-quality chinese datasets for llm training. *Preprint*, arXiv:2501.08197.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12360–12371.

Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Y. Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhu Chen. 2024a. Map-neo: Highly capable and transparent bilingual large language model series. *CoRR*, abs/2405.19327.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024b. TinyLlama: An Open-Source Small Language Model. *arXiv preprint*. ArXiv:2401.02385 [cs].

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang

Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yutao Zhu, Kun Zhou, Kelong Mao, Wentong Chen, Yiding Sun, Zhipeng Chen, Qian Cao, Yihan Wu, Yushuo Chen, Feng Wang, Lei Zhang, Junyi Li, Xiaolei Wang, Lei Wang, Beichen Zhang, Zican Dong, Xiaoxue Cheng, Yuhan Chen, Xinyu Tang, Yupeng Hou, Qiangqiang Ren, Xincheng Pang, Shufang Xie, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, Yankai Lin, Ruihua Song, Jun Xu, Xu Chen, Rui Yan, Zhewei Wei, Di Hu, Wenbing Huang, Ze-Feng Gao, Yueguo Chen, Weizheng Lu, and Ji-Rong Wen. 2024. Yulan: An open-source large language model. *CoRR*, abs/2406.19853.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022a. Designing effective sparse expert models. *CoRR*, abs/2202.08906.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022b. ST-MoE: Designing Stable and Transferable Sparse Expert Models. (arXiv:2202.08906).

A Overall Pre-Training Configuration

In this section, we will provide an overview of the pre-training configuration, introducing its key components and the algorithms involved in the process. For a more detailed discussion of the major contributions made in this work, please refer to Section 3.2 and Section 4.

A.1 Model Architecture

Our model is based on a decoder-only transformer with a tall and narrow architecture, inspired by previous studies (Liu et al., 2024c; Hu et al., 2024). It comprises a total of 2.42B parameters, of which 2.23B are non-embedding parameters. The hyper-parameter configurations for our model architecture are provided in Table 6. Additionally, we reparameterize each weight matrix of different modules with an extra learnable parameter (Nishida et al., 2024), enhancing the model's training stability (discussed in Section 3.2). Next, we briefly introduce the main components in our architecture.

Embedding tying We utilize embedding tying (Press and Wolf, 2017) to reduce the model's parameter size and stabilize training. In our preliminary experiments, we find that sharing the embedding and unembedding matrices improves model convergence. Furthermore, when these matrices are not shared, they often necessitate different initialization strategies, which we will discuss in Section 3.2.

Pre-RMSNorm Layer normalization (LN) has been shown to enhance numerical stability and accelerate learning speed (Ba et al., 2016). We integrate Pre-LN into our model architecture to improve convergence stability and speed compared to Post-LN (Xiong et al., 2020). Regarding the form of normalization, we opt for `RMSNorm` over the conventional `LayerNorm`, as it conserves CUDA memory while attaining a comparable effect (Zhang and Sennrich, 2019).

SwiGLU Our model introduces non-linearity using a gated linear unit (GLU) with the Swish activation function, known as SwiGLU (Shazeer, 2020). This method effectively captures complex data relationships and has proven to be effective in relatively small language models, as demonstrated by (Liu et al., 2024c).

Attention mechanism We adopt the grouped-query attention (GQA, (Ainslie et al., 2023)), which enables the model to reduce KV cache usage while maintaining high performance. Specifically, we employ 30 heads for query attention and 6 groups for key-value heads. We opt not to make the KV head size divisible by 8 since small language models rarely require tensor parallelism during inference. We only add bias for QKV projections.

Rotary Embedding We adopt rotary positional embedding (RoPE) to capture the positional information in our model, since it integrates absolute and relative positioning in an unified way. During the stable training stage, we set the parameter θ to 10,000, and increase it to $49\,000$ during the annealing stage to extend the context length to 28,672 (28K) tokens using adjusted base frequency (ABF).

A.2 Tokenizer

Tokenization is a critical preprocessing step that splits input text into sequences of tokens. Below, we provide details of our tokenizer.

Vocabulary size Generally, the vocabulary size should be chosen to balance its effects on the model's parameter size and efficiency. We adopt the three approaches proposed by (Dagan et al., 2024) to balance the compute budget and vocabulary capacity, yielding a final vocabulary size of around 99,000. For simplicity, we reuse the Byte Pair Encoding (BPE) tokenizer of MiniCPM (Hu et al., 2024). Specifically, we truncate the vocabulary by applying the corresponding BPE merge rules to reduce the number of tokens. We also heuristically remove rare domain-specific tokens, while add some reserved tokens in the vocabulary. The statistics of the modified vocabulary and the compression rate are shown at Table 7. The test set for the tokenization experiments is sourced from a diverse array of datasets, as detailed in Section B.4. Overall, our tokenization method achieves a well-balanced compression rate across different domains.

BPE-dropout Existing sub-word tokenization methods prevent the language models from understanding the alphabetic composition of a token. To mitigate this issue, BPE-dropout (Provilkov et al., 2020) has been proposed to help the model better learn the internal representation of a token, enabling it to more effectively capture possible subwords within a word. Specifically, we use a relatively low dropout rate of 0.2, and applying the dropout method results in only a slight increase in the number of tokens (0.07%), as shown in Table 7.

Digit tokenization Digit tokenization plays a crucial role in mathematical tasks, including numerical calculation and complex reasoning. We follow the common practice of splitting numbers into individual digits (Bi et al., 2024; Yang et al., 2023). Although other methods, such as three-digit tokenization, may achieve higher compression rates, using individual-digit tokenization typically leads to improved numerical calculation accuracy (Wang et al., 2024).

A.3 Training Data Preparation

Data serves as the foundation for developing the model's capabilities, and we employ specially designed strategies for collecting and preparing the training dataset. Next, we briefly describe the general procedure for data preparation. A more detailed and comprehensive description of the data pipeline is provided in Section 4.

Data collection and selection To ensure reproducibility, our pre-training data is primarily sourced from open-source pretraining datasets and synthetically generated data. The main open-source datasets include FineWeb-Edu (Lozhkov et al., 2024a), the-stack-v2 (Lozhkov et al., 2024b), open-web-math (Paster et al., 2024), Chinese-FineWeb-Edu (**?**), and OpenCoder-LLM (Huang et al., 2024). The entire pre-training dataset has undergone rigorous preprocessing, with 1.08T tokens for training. Among them are 481B English web data, 138B general English knowledge, 227B code pre-training data, 16.7B code instruction data, 93.8B mathematics pre-training data, 15.5B mathematics instruction data, and 108B Chinese data.

Data schedule Using the WSD scheduling method (Hu et al., 2024), the training process is divided into three main stages: warmup, stable training, and annealing. The warmup stage uses 10B tokens, the stable training stage utilizes 990B tokens, and the annealing stage uses 80B tokens. To better manage the training process, we divide the entire training trajectory into 27 consecutive *curriculum phases*, each consisting of 40B tokens. When transitioning between these curriculum phases, the dataset proportions are slightly adjusted based on the model's performance on various benchmarks and the perplexity (PPL) of validation texts. However, the internal data distribution of each curriculum phase cannot be modified once it has been

Table 6: Hyperparameter settings of diffrent models. r_{ffn} is the ratio of the feed-forward network's hidden size to the model's hidden size. The definition of the symbols is available at Table 8

Model	n_{layers}	d_{model}	r_{ffn}	n_{heads}	$n_{\text{kv\_heads}}$
LLaMA-3.2-3B	28	3,072	2.7	24	8
Phi-3-mini-4k-instruct	32	3,072	2.7	32	32
MiniCPM-2B	40	2,304	2.5	36	36
MiniCPM3-4B	62	2,560	2.5	40	40
Qwen2.5-1.5B	28	1,536	5.8	12	2
MobileLLM-1B	54	1,280	2.8	20	5
YuLan-Mini	56	1,920	2.5	30	6

Table 7: Compression rate of different tokenizers. Higher values indicate more effective compression.

Tokenizer	**Vocabulary Size**	**Web**	**Chinese**	**Math**	**Code**
Gemma2-2B	256,000	4.928	3.808	2.865	3.309
Qwen2.5	151,936	<u>4.935</u>	3.956	2.890	<u>3.881</u>
LLaMA-3.1	128,000	**4.994**	3.263	**3.326**	**3.911**
MiniCPM-2.4B	122,753	4.753	**4.273**	<u>2.739</u>	3.052
Phi-3.5-mini	100,352	4.311	1.914	2.654	3.110
MiniCPM-1.2B	73,440	4.631	4.042	2.696	3.017
YuLan-Mini	99,000	4.687	<u>4.147</u>	2.716	3.033
+ Dropout	99,000	4.687	4.146	2.715	3.031

scheduled for training. During the annealing stage, the proportion of instruction data and long context data is increased. Following the work by (Hu et al., 2024), we estimate the optimal annealing ratio to be 8%, *i.e.,* 80 billion tokens. We maintain the same batch size used during stable training, *i.e.,* 4 million tokens. The learning rate is decreased from 10^{-2} to 5.22×10^{-5} over a span of 18,802 steps. Subsequently, the learning rate is held constant at 5.22×10^{-5} for the final 772 steps.

A.4 Model Optimization

For model optimization, hyperparameters are crucial for training stability and model performance.

Specifically, we adopt the WSD learning rate scheduler (Hu et al., 2024). Maintaining a constant learning rate during the stable training stage eliminates the necessity to specify an ending step, as required by the cosine scheduler. This approach facilitates continuing pre-training from the last checkpoint during stable training. It also allows for more flexible data preparation: we can prepare the data while the preceding curriculum phase is running. Additionally, we estimate an optimal annealing ratio of 8% for the stable training stage using the scaling law of learning rate annealing (Tissue et al., 2024).

For training stability, we combine a parameter initialization approach akin to μP (Dey et al., 2023b; Hu et al., 2024; Yang et al., 2022) with WeSaR re-parameterization (Nishida et al., 2024), using a relatively large global learning rate of 0.01. The rationale behind adopting a large learning rate is the expectation that the model will possess greater potential for enhancement during the annealing stage. We set the AdamW hyperparameters as follows: $\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-15}$, with the `weight_decay` of 0.1 and the `z-loss` coefficient of 10^{-4} (de Brébisson and Vincent, 2016). We use a variance of 5×10^{-5} for initialization. As found by (Wortsman et al., 2024), extending the warm-up ratio enhances training stability, so we linearly warm up the model over 10B tokens. We use a batch size of 4.12M tokens with a sequence length of 4,096, extending the context length during the annealing stage while keeping the total token count in the batch size unchanged. We avoid using gradient accumulation to prevent numerical precision error of `bfloat16`. Detailed analysis of training stability can be found in Section 3.2.

A.5 Training Infrastructure

We build a simple yet efficient training framework based on the HuggingFace `Trainer` and other open-source libraries (`DeepSpeed`, `flash-attention`, and liger-kernel).

Specifically, we first use ZeRO-1 (Rajbhandari et al., 2020) data parallelism provided by DeepSpeed intergration and then switch to ZeRO-2 after confirming that it does not cause training divergence in our model.[2] We also leverage Flash Attention (Dao et al., 2022; Dao, 2024) and a triton kernel library liger-kernel (Hsu et al., 2024) to accelerate training processes. By employing fused kernels, we achieve a 30% reduction in training time and up to 70% savings in CUDA memory.[3] We further optimize the balance between CUDA memory usage and training time by adjusting the number of layers through the activation checkpointing function. For enhanced training efficiency, we use bfloat16 precision for both model parameters and NCCL communications. The model's FLOPs utilization (MFU) is estimated at 51.57%.

Regarding the hardware setup, we initially employ a 56 A800-GPU cluster managed by the SLURM system (Yoo et al., 2003). We later reduce the number of GPUs to 48 by transitioning the distributed optimizer to a universal checkpoint (Lian et al., 2024). To maximize device utilization, we perform tokenization and packing asynchronously. Given the modest size of our cluster, the likelihood of encountering NCCL failures is relatively low. Therefore, after assessing the advantages and disadvantages, we decide to store a checkpoint every hour and implement automatic restarts.

For efficient evaluation, we utilize LLMBox (Tang et al., 2024) to integrate vLLM (Kwon et al., 2023) for generative tasks and employ KV cache scheduling for multiple-choice tasks. For a detailed description of the evaluation setup and results, please refer to Appendix B.

A.6 Long Context

Previous research (Chen et al., 2023) has demonstrated that LLMs can hardly process texts exceeding their context windows due to the out-of-distribution (OOD) rotation angles in RoPE. To achieve the context window extension, increasing the base frequency of RoPE to migrate the OOD rotation angles and continual pre-training has been an effective method (Xiong et al., 2024). Consequently, during the annealing stage, we increase the base frequency of RoPE θ from 10,000, employed during stable training, to 490,000 and train the model on long texts. This adjustment successfully extends the context length from 4,096 (4K) tokens to 28,672 (28K) tokens.

During the annealing stage of the final 80B tokens, we adjust the base frequency of RoPE from 10,000 to 490,000 and train on long sequences to extend the context length from 4,096 tokens to 28,672 tokens. We avoid training with long contexts in earlier stages because the computational cost of self-attention layers increases quadratically with sequence length, making it prohibitively expensive (Dubey et al., 2024).

When training on long contexts, we observe a decline in the model's performance on short-text benchmarks. To enhance the long-text capacities and preserve the short-text capacities, we carefully design the mixture of data. We upample books and concatenated GitHub code texts (Liu et al., 2024b) as long context data to capture long-term dependencies, while using high-quality short texts to preserve short-text capabilities. Additionally, inspired by previous studies (Ding et al., 2024; Gao et al., 2024), we also apply masked cross-document attention that prevents attention across different documents to preserve short-context capabilities.

A.7 Other Strategies

Packing Since the training data during the annealing stage includes some instruction data, using a traditional simple packing method for pre-training data could result in instruction data being split, thereby compromising its effectiveness. To address this, we propose a packing strategy designed to maintain training efficiency while minimizing the disruption of instruction data. This strategy involves different packing methods based on data type. Pre-training data is directly spliced, whereas for instruction data, if it is divided into two sequences, the remaining part of the previous sequence is padded directly, and this instruction data serves as the beginning of the second sequence. Subsequently, any redundant padding tokens are replaced with pre-training data tokens. By including the instruction data, our main goal is to learn the reasoning process rather than focusing solely on the question-and-answer format. Therefore, we employ the same data processing method used in pre-training, which directly includes question-answer pairs without relying on a chat template. When calculating the loss, the instruction and response are treated as a single document, and the loss for

[2] https://github.com/microsoft/DeepSpeed/issues/6351

[3] Fused kernels include: SelfAttention, RMSNorm, RoPE, SwiGLU, FusedLinearCrossEntropy, and AdamW. torch.compile is also enabled in our implementation.

the instruction is not masked.

Checkpoint merging Following the approach used in LLaMA3 (Dubey et al., 2024), we combine the last few checkpoints during the annealing stage to produce the final pre-trained model. While this strategy might result in a slight reduction in certain specific capabilities (*e.g.,* GSM8K), it generally leads to a more well-rounded model.

B Experimental Setup

B.1 Evaluation Benchmarks

For a comprehensive evaluation of LLMs performance, we select the benchmarks from the following aspects.

- *Language comprehension*: We select the widely-used English benchmarks MMLU (Hendrycks et al., 2021a), LAMBADA (Kazemi et al., 2023) and RACE (Lai et al., 2017), along with the Chinese benchmarks CMMLU (Li et al., 2024a) and CEval (Huang et al., 2023), to evaluate the bilingual comprehension capabilities of the LLM. These benchmarks span various domains, such as history, science, and culture.

- *Code generation*: We select Humaneval (Chen et al., 2021) and MBPP (Austin et al., 2021) to assess the capability of LLMs to generate accurate code snippets for natural language problems.

- *Mathematical reasoning*: We utilize GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021b; Lightman et al., 2024) to evaluate the mathematical reasoning capabilities of LLMs. These benchmarks range from basic arithmetic to advanced mathematical problems.

- *Logical reasoning*: We assess the logical reasoning capabilities of LLMs using ARC-E (Yadav et al., 2019), ARC-C (Yadav et al., 2019), which provide a comprehensive evaluation of logical reasoning across various knowledge domains.

- *Commonsense reasoning*: We evaluate the LLM's commonsense reasoning ability using WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), StoryCloze (Mostafazadeh et al., 2016) which test the understanding and utilization of daily commonsense knowledge.

B.2 Baseline Models

To ensure a comprehensive evaluation, we select several small LLMs with comparable scales (*i.e.,* base models ranging from 0.5 to 3B, including embedding sizes) as baselines for comparison:

- *MiniCPM-2.4B* (Hu et al., 2024): MiniCPM-2.4B is pre-trained on 1.06T tokens and also employs the annealing training strategy. Despite its small size (2.7B total model size), it exhibits impressive performance in general tasks while supporting deployments with limited hardware resource.

- *Qwen series models* (Qwen-Team, 2024; Yang et al., 2024a): We select Qwen2-1.5B, Qwen2.5-0.5B, and Qwen2.5-1.5B for comparison. The latest Qwen2.5 series of small LLMs have been pre-trained on 18T tokens, and the training details have not been fully publicly released. They demonstrate state of the arts performance in both general and domain-specific tasks.

- *StableLM2-1.6B* (Bellagente et al., 2024): StableLM2-1.6B is a small LLM proposed by StabilityAI. It has been pre-trained on a mixture of open-source datasets, which utilizes several small LLMs to determine the training data proportion.

- *SmolLM2-1.7B* (Allal et al., 2024): SmolLM2-1.7B is developed by HuggingFace TB Research based on its collected high-quality pre-training corpus, which has been trained on 11T tokens, and maintains a good balance between speed and accuracy.

- *Llama3.2-3B* (Dubey et al., 2024): Llama3.2-3B (3.2B total model size) is developed by MetaAI, which is trained on up to 9T tokens. It further distills the knowledge from LLaMA3.1-8B and 70B models by using their logits during the pre-training stage.

- *Gemma2-2.6B* (Gemma Team, 2024): Gemma2-2.6B is developed by Google, which is trained on 2T tokens, mainly including web documents, code, and mathematical text.

- *Falcon3-3B* (Falcon-LLM Team, 2024): Falcon3-3B is a transformer model initialized from Falcon3-7B-Base by pruning with further distillation to recover using 1024 H100 GPU chips.

B.3 Implementation Details

To comprehensively compare the performance of different LLMs, we employ diverse evaluation settings and design specific methods for guaranteeing the fairness and efficiency.

- *Zero-shot and few-shot settings:* Following existing work (Qwen-Team, 2024), For LAMBADA, HumanEval, MBPP, RACE, StoryCloze and RULER, we adopt the zero-shot setting. For GSM8K and MATH, we adopt the 4-shot setting. For MMLU, CMMLU, WinoGrande and CEval, we adopt the 5-shot setting. For HellaSwag, we adopt the 10-shot setting. For ARC-E, ARC-C, we adopt the 25-shot setting.

- *Chain-of-Thought (CoT):* For GSM8K and MATH, we follow previous work (Qwen-Team, 2024) that uses CoT prompting to facilitate the LLM to perform step-by-step reasoning. Considering the potential performance variance caused by CoT prompts, we utilize both the short ones provided by the original dataset and the long ones generated by `kimi-k0-math`. For each model, we evaluate the performance using both prompt types, and select the one yielding the higher score as the result.

- *Evaluation metrics:* For QA tasks, we employ `maj@1` for GSM8K and MATH, `pass@1` for HumanEval and MBPP, and accuracy of the model response for remaining generation tasks. For multiple-choice questions, we primarily evaluate based on the accuracy of the generated answer, which is determined by selecting the choice with the lowest perplexity. However, for ARC-E and ARC-C, we utilize normalized accuracy (Brown et al., 2020). performance of MATH-500, we further use `gpt-4o-mini` to verify the correctness of the results generated by all models and conducted manual checks.

- *Maximum length:* For GSM8K and MATH, since CoT prompting may result in longer outputs, we set the maximum generation length to 596 for short context (*i.e.,* 4K) models and 2,048 for long context models. For HumanEval and MBPP, we set the maximum generation length to 400. For other generative tasks, we set it to 128 for efficiency.

- *Evaluation framework:* For the majority of tasks, we employ LLMBox (Tang et al., 2024) to assess performance. Specifically, for generation tasks, we enable vLLM (Kwon et al., 2023). However, to ensure reproducibility, we utilize EvalPlus (Liu et al., 2024a) for HumanEval and MBPP.

Despite our considerable efforts, fully reproducing the results of these baseline models as originally reported remains challenging, due to the lack of detailed evaluation setup information. For a fair comparison, we report the performance results of the baselines as provided in their official technical reports.

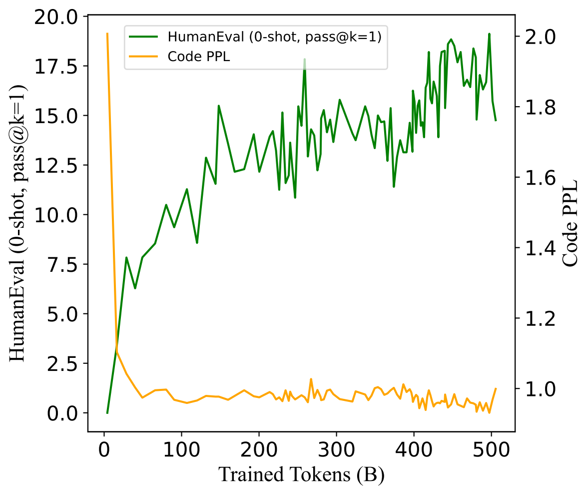
B.4 Evaluating Model Performance during Pre-Training

During pre-training, it is crucial to continuously evaluate the model's performance to monitor for any unstable or abnormal training issues. However, existing benchmarks rely on advanced abilities (*e.g.,* instruction following), which often develop with sufficient data training. Thus, the model's performance tends to remain at a low level on these benchmarks in the early stages, and directly evaluating the model's performance on specific validation sets would not provide an accurate assessment.

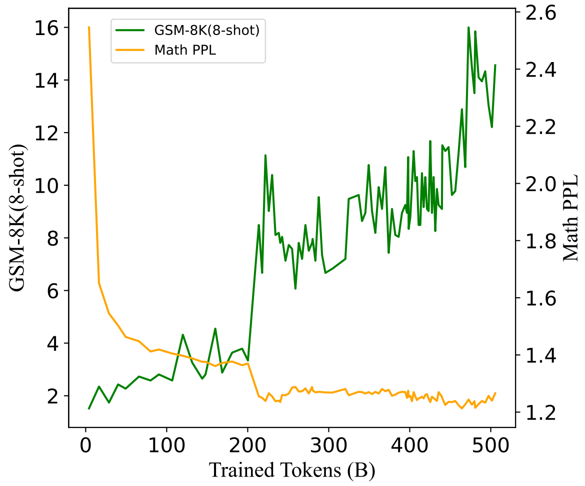
To address this, we have designed two monitoring strategies for different stages of training. In the early stages, we assess the model's performance primarily through perplexity measures on the constructed validation datasets and LAMBADA benchmark. In the later stages, we shift to using performance on selected benchmarks (*e.g.,* HumanEval and GSM8K) for more comprehensive evaluation. Next, we introduce how to construct the validation set for perplexity measurement at early stage of pre-training.

To comprehensively evaluate the key abilities of our model, we create four validation sets from the following aspects, namely English understanding, Chinese understanding, code generation, and math reasoning. The detailed data composition is as follows.

- *English understanding*: We randomly select



(a) Performance curve on HumanEval.



(b) Performance curve on GSM8K.

Figure 11: Performance comparison using perplexity (PPL) and accuracy-based metrics to monitor the code generation and math reasoning abilities of YuLan-Mini.

2,118 samples from FineWeb-Edu and compute the perplexity for ability evaluation.

- *Chinese understanding*: We randomly select 1,679 samples from Chinese-FineWeb-Edu for computing the perplexity.
- *Code generation*: We randomly select 2,067 samples from a widely-used code instruction datasets, Python-Code-Instructions-18k-Alpaca for perplexity evaluation.[4]
- *Math reasoning*: We randomly sample 1,499 open-ended questions from MathInstruct (Yue et al., 2024) for perplexity.

[4] https://huggingface.co/datasets/iamtarun/python_code_instructions_18k_alpaca

Once the advanced capabilities are well-developed, we can directly monitor the model's performance by evaluating it on the selected benchmarks.

Training setup Since it is resource-intense to perform extensive experiments on our model, we explore the training dynamics by conducting surrogate experiment with a small proxy model of 0.2B with similar architecture. We employ a relatively large learning rate of 0.01, to expose potential instabilities within the model. We keep this baseline model setup in the subsequent experiment, which we elaborate on in Appendix C. Specifically, our optimization goal is to achieve optimal performance while ensuring that the training process does not result in divergent loss or an increasing trend in gradient norm.

C Training Stability

C.1 Indicators Setup

In large-scale training, distributed optimizers are often used, which means that the gradients of different modules may be distributed across various data parallel ranks. This distribution makes it inefficient to directly obtain the gradients. As a result, we primarily track each module's weight matrix and hidden states (*i.e.,* their outputs). Specifically, we record the mean and variance of the weights and hidden states, as well as the root mean square (RMS), which is calculated using the follow formula RMS $= \sqrt{\mathrm{Var} + \mathrm{Mean}^2}$. Note we consider the outputs of various modules in the transformer (*i.e.,* `FFN`, `Attention`, `RMSNorm`) as hidden states.

C.2 Exploding Hidden States Due to Residual Connection

Here we provide a detailed derivation for Equation 3, which aims to investigate the growing hidden states due to residual connection. To understand the underlying cause, we express the hidden states in terms of the model's weights and inputs:

$$\mathrm{var}(\boldsymbol{z}^l) = \mathrm{var}(\boldsymbol{y}^l) + \mathrm{var}(\texttt{FFN}(\texttt{RMSNorm}(\boldsymbol{y}^l))),$$
$$\mathrm{var}(\boldsymbol{y}^l) = \mathrm{var}(\boldsymbol{x}^l) + \mathrm{var}(\texttt{MHA}(\texttt{RMSNorm}(\boldsymbol{x}^l))).$$

For ease of analysis, we first assume that:

$$\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

Table 8: Definition of the variables for computing the hyperparameters.

Variables	Meaning
n_{layers}	The num of model's layers, *i.e.,* `num_hidden_layers`.
n_{heads}	The num of model's attention heads, *i.e.,* `num_attention_heads`.
$n_{\text{kv\_heads}}$	The num of model's kv-heads used in GQA, *i.e.,* `num_key_value_heads`.
d_{model}	Model dimension, *i.e.,* `hidden_size`.
d_{head}	Dimension of attention head, *i.e.,* `hidden_size / num_attention_heads`.
d_{ffn}	The hidden size of feed-forward network, *i.e.,* `intermediate_size`.
σ_{base}	Initialization standard deviation for each matrix, *i.e.,* `initializer_range`.
η_{base}	Learning rate, *i.e.,* max learning rate.
$\odot$	Element-wise multiplication.
FFN	SwiGLU $\mathtt{FFN}(\boldsymbol{u}) = [\mathcal{F}(\boldsymbol{u}\mathbf{W}_{\text{gate}}) \odot (\boldsymbol{u}\mathbf{W}_{\text{up}})]\mathbf{W}_{\text{down}}$, where SiLU $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{x} \odot \sigma(\boldsymbol{x})$.
RMSNorm	Root mean square layer normalization without bias $\mathtt{RMSNorm}(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\text{RMS}(\boldsymbol{x})} \odot \boldsymbol{g}$.
MHA	Multi-head attention $\mathtt{MHA}(\boldsymbol{v}) = \text{concat}_{i=1}^{h}[\mathtt{head}_i(\boldsymbol{v})]\mathbf{W}_o$.
head($\mathbf{X}$)	$\mathtt{head}(\mathbf{X}) = \text{Softmax}(\frac{\mathbf{S}}{\sqrt{d_{heads}}})\mathbf{X}\mathbf{W}_{\mathbf{V}}$, where the attention weights $\mathbf{S} = \mathbf{X}^T\mathbf{W}_Q^T\mathbf{W}_K\mathbf{X}$.
$d_{\text{model\_proxy}}$	d_{model} for proxy model, *i.e.,* the 0.05B model
m_{width}	Width scaling factor in μP, *i.e.,* $d_{\text{model}}/d_{\text{model\_proxy}}$

Under this assumption, we can obtain $\text{var}(\boldsymbol{u}) = \text{var}(\boldsymbol{v}) = 1$. In this case, we can express the variance as the following form:

$$\text{var}(\boldsymbol{z}^l) = \text{var}(\boldsymbol{x}^l) + \text{var}(\mathtt{FFN}(\boldsymbol{u})) + \text{var}(\mathtt{MHA}(\boldsymbol{v})), \tag{5}$$

which means, the hidden states will grow by the variance of MHA and FFN in each layer:

$$\text{var}(\mathtt{head}_i(\boldsymbol{v})) = \text{var}(\text{softmax}(\mathbf{Z})\mathbf{V}) \cdot d_{\text{model}} \cdot \text{var}(\mathbf{W}_v) < d_{\text{model}} \cdot \text{var}(\mathbf{W}_v), \tag{6}$$

$$\text{var}(\mathtt{FFN}) = d_{\text{ffn}} \cdot d_{\text{model}} \cdot \text{var}(\mathbf{W}_{up}) \cdot \text{var}(\mathbf{W}_{down}), \tag{7}$$

$$\begin{aligned}\text{var}(\mathtt{MHA}) &= \text{var}(\mathtt{head}(\boldsymbol{v})) \cdot d_{\text{model}} \cdot \text{var}(\mathbf{W}_o) \\ &< d_{\text{model}}^2 \cdot \text{var}(\mathbf{W}_v) \cdot \text{var}(\mathbf{W}_o),\end{aligned} \tag{8}$$

where $\mathbf{Z}$ denotes the scaled attention scores. The base dimensionality d_{model} of LLMs are often large (*e.g.,* 1,920 in our model).

Therefore, the variance addition of each layer $\Delta H^l = \text{var}(\boldsymbol{z}^l) - \text{var}(\boldsymbol{x}^l) = \text{var}(\mathtt{MHA}(\boldsymbol{v})) + \text{var}(\mathtt{FFN}(\boldsymbol{u}))$. By plugging in Equation 7 and 8, we can estimate the upper bound of ΔH^l as:

$$\Delta H^l < d_{\text{model}}^2 \cdot \text{var}(\mathbf{W}_v) \cdot \text{var}(\mathbf{W}_o) + d_{\text{ffn}} \cdot d_{\text{model}} \cdot \text{var}(\mathbf{W}_{up}) \cdot \text{var}(\mathbf{W}_{down}). \tag{9}$$

C.3 Discussion on Other Training Stabilization Methods

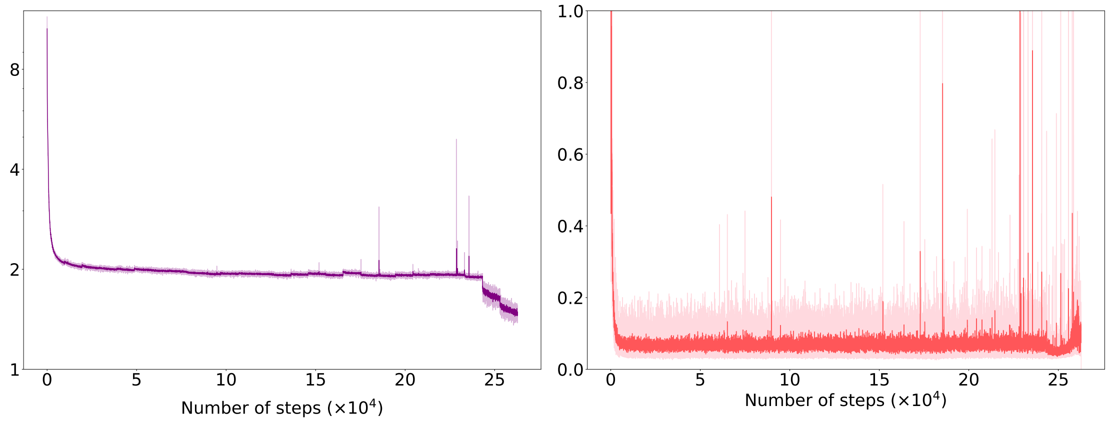
During our training process, we thoroughly explore and utilize various training stabilization techniques. Below, we provide a brief introduction to these methods.

C.3.1 Warmup Based Methods

To ensure the model transitions smoothly from its initial state to a stable training phase, we empirically find that employing learning rate warmup and sequence length warmup is often effective, which are detailed below.

Learning rate warmup Learning rate warmup involves gradually increasing the learning rate from a small initial value (*e.g.,* 0) to the max learning rate in T_{LR} steps. (Wortsman et al., 2024) suggests that a longer learning rate warmup can reduce sensitivity to the learning rate, as measured by training stability across different learning rates. We empirically verify this conclusion and find increasing T_{LR} indeed enhances training stability. For our final training, we set $T_{\text{LR}} = 2{,}433$, which approximately corresponds to 10 billion tokens of data.

Sequence length warmup Sequence length warmup starts training with short sequences (*e.g.,* 64 tokens) and gradually increases their length within the steps of T_{SL}, which is typically set to a



(a) Training loss.

(b) Gradient norm.

Figure 12: Training loss and gradients during pre-training process.

Table 9: Comparison of the used hyperparameter settings for training stability, where the detailed explanation for the variables are in Table 8. We include SI (Takase et al., 2023) for comparison, MiniCPM (Hu et al., 2024), CerebrasGPT (Dey et al., 2023a). The definition of the symbols is available at Table 8 .

Method	**SI**	**MiniCPM**	**CerebrasGPT**	**YuLan-Mini**
Scale Embedding Output	1	12	10	10
Scale MHA equation	$1/\sqrt{d_{\text{head}}}$	$1/\sqrt{d_{\text{head}}}$	$1/d_{\text{head}}$	$1/\sqrt{d_{\text{head}}}$
Scale Residual Connection	1	$\frac{1.4}{\sqrt{n_{\text{layers}}}}$	1	$\frac{1.4}{\sqrt{n_{\text{layers}}}}$
QKV Weights LR	η_{base}	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$
QKV σ Init	σ^2_{base}	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$
O Weights LR	η_{base}	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$
O σ Init	$\frac{\sigma^2_{\text{base}}}{2n_{\text{layers}}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\frac{\sigma^2_{\text{base}}}{2m_{\text{width}}\cdot n_{\text{layers}}}$	$\frac{\sigma^2_{\text{base}}}{2m_{\text{width}}\cdot n_{\text{layers}}}$
FFN1 Weights LR	η_{base}	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$
FFN1 σ Init	σ^2_{base}	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$
FFN2 Weights LR	η_{base}	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$	$\eta_{\text{base}}/m_{\text{width}}$
FFN2 σ Init	$\frac{\sigma^2_{\text{base}}}{2n_{\text{layers}}}$	$\sigma^2_{\text{base}}/m_{\text{width}}$	$\frac{\sigma^2_{\text{base}}}{2m_{\text{width}}\cdot n_{\text{layers}}}$	$\frac{\sigma^2_{\text{base}}}{2m_{\text{width}}\cdot n_{\text{layers}}}$
Scale Output logits	1	$1/m_{\text{width}}$	$1/m_{\text{width}}$	1

few multiples of T_{LR} (Li et al., 2022). The rationale behind this approach is that longer sequence lengths contribute significantly to extreme gradient variance, particularly in the early stages of training. In our experiments, we also observe similar fluctuations in loss during long context training (especially in the 27-th curriculum phase). However, since we have stabilized the training using other methods and this approach requires additional preparation of the data, we ultimately decided not to adopt it.

C.3.2 Module Based Methods

In this part, we introduce module-based methods which regularize the model states by adjusting specific components in it.

QK LayerNorm QK LayerNorm and its variants (*e.g.,* QKV LayerNorm or capped QK LayerNorm) have have been shown to effectively mitigate the growth of attention logits (Rybakov et al., 2024), which we also have identified in Section 6.1. We highlight the effectiveness of QK LayerNorm because it directly addresses the exponential growth of gradients caused by the interaction of hidden states ($\mathbf{QK}^T$), whereas some other methods only attempt to control the downstream instability. Our empirical study, which is shown in Figure 13a and 13b, demonstrates the advantages of QK LayerNorm in terms of training stability. However, it significantly slows down the calculation in training: with the same acceleration configuration, using QK LayerNorm increases the training time by 34%.

Note that the implementation of QK LayerNorm here is similar to StableLM's per-head approach, allowing each attention head to learn independently. Considering that the previously mentioned methods have already demonstrated stability in our preliminary experiments, we ultimately decided not to use QK LayerNorm (Section 6.1).

Embedding tying Embedding tying aims to share the weights of embedding and unembedding (*i.e.,* `lm_head`) parameters (Press and Wolf, 2017). Our experiments demonstrate that the utilization of embedding sharing enables faster convergence and more stable training, and there is no significant degradation in training performance.

Z-loss Z-loss was originally proposed to alleviate the shift and scale of logits in classification tasks (de Brébisson and Vincent, 2016). Subsequently, it has been introduced to LLM and MoE training to mitigate the growth of the logits layer (Chowdhery et al., 2023; Zoph et al., 2022a). It adds an auxiliary term related to the softmax normalizer $\log Z$ to the original loss: $\mathcal{L} = \texttt{lm\_loss} + \zeta \log^2 Z$. In our experiments, we set the coefficient $\zeta = 10^{-4}$ to encourage the logits to be close to 0. Although ablation studies did not show significant effects, we incorporate it into the final training.

C.3.3 Numerical Optimization Based Methods

In addition, we consider using several commons methods to reduce abnormal updates during optimization, as described below.

Weight decay To prevent abnormal model weights due to large gradient updates, weight decay functions by subtracting a penalty term from the weights during the update step, rather than directly modifying the gradients. Formally, we denote the AdamW update without learning rate or weight decay as:

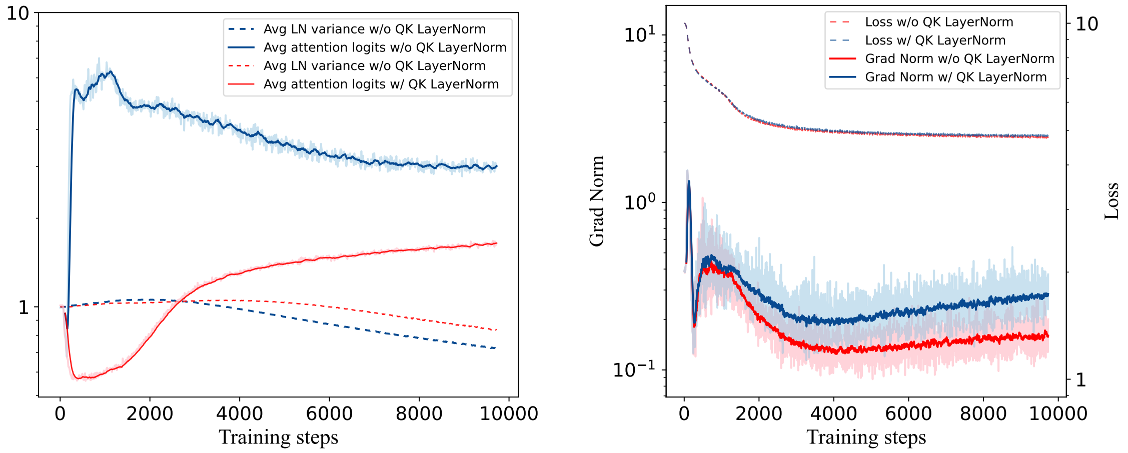
$$\Delta = \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon). \tag{10}$$

Then at update step t, the AdamW update with weight decay is given by $\theta \rightarrow \theta - s_t \eta(\Delta - \lambda\theta)$, where λ is the weight decay coefficient, s_t is learning rate schedule and η is the max learning rate. Previous work has recommended using an independent weight decay for updates, expressed as $\theta \rightarrow \theta - s_t(\eta\Delta - \lambda'\theta)$, which is claimed to be applicable to a wider range of learning rates (Loshchilov and Hutter, 2019; Wortsman et al., 2024). In the PyTorch implementation, this approach can be achieved by tuning the weight decay coefficient λ in conjunction with the maximum learning rate, following the relationship $\lambda' = \eta \cdot \lambda$.

Optimizer hyper-parameter In the update of AdamW (Equation (10)), $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{v}}_t$ represent the first and second gradient moment exponential moving averages (EMA), respectively. If the gradient is of the same order of magnitude as ϵ, then the update value Δ will be significantly reduced due to ϵ, which empirically leads to training instability inherent in embedding layer. A direct solution is to reduce ϵ from the default value of 10^{-8} to 10^{-15}. Generally speaking, this method can alleviate the divergence caused by abnormal embedding gradient values in larger-scale models (Wortsman et al., 2024; Molybog et al., 2023).

Numerical stability In practice, paying close attention to numerical stability is crucial, as it can be an important source of training instability. In large-scale model training, `float32` often suffers from low computational efficiency. Although `float16` offers comparable precision with higher computational efficiency, it has a limited numerical representation range (*e.g.,* maximum positive number that can be represented is 65,504). Therefore, `bfloat16` has been proposed as a trade-off between precision and representation range. It largely alleviates the training instability caused by exceeding the representable range. However, in practice, `bfloat16` introduces precision problems compared to `float16`. In experiments conducted by (Lee et al., 2024) using `bfloat16` with 188 random seeds, 18 runs diverged, whereas using `float32` under the same configuration resulted in all runs converging normally. To mitigate precision issues with `bfloat16`, Gemma (Mesnard et al., 2024) find that shifting the RMSNorm weight from 1 to 0 helps, considering that `bfloat16` has symmetric numerical precision around 0 but greater inaccuracies near 1.

Value clipping To further limit the gradient within certain range, we utilize a gradient clipping of 1. We find using a smaller limit does not help stabilize the training. In addition, initializing the LLM in accordance with "3-σ" rule with `nn.init.trunc_normal_` may be helpful for numerical stability.



(a) Variance of attention values and LN outputs

(b) Gradient norm and loss trajectory

Figure 13: The curves of attention value and LN output variances (left) and gradient norm and loss (right). After using QK LayerNorm, we prevent the explosion of attention logits and gradients, keeping the LN output stable around 1 and the loss consistent.

D Data Filtering Pipeline

As we aim for a data-efficient training approach, data quality is crucial to the final model's performance. For this purpose, we implement a thorough data cleaning process to remove low-quality texts (Figure 6).

De-duplication Data de-duplication is a crucial step in standard LLM training practices, as previous research has demonstrated that duplicate data can significantly degrade model performance (Tirumala et al., 2023). We use the MinHash algorithm implemented by the Yulan-GARDEN library (Sun et al., 2024) to deduplicate the training data.

Heuristic filtering We adopt heuristic methods to filter the data, some of which are listed as follows:

- All: we remove the documents containing fewer than 20 tokens.
- Code: we apply filtering criteria based on code metrics (*e.g.,* average line length, alphabetic characters ratio, and keyword statistics) similar to DeepSeek-Coder (Guo et al., 2024).
- Synthetic data: we remove responses that are garbled or contain repeated content. For math texts, we remove response that do not contain an hightlited answer part (*e.g.,* `$box{}$`).

Topic-based text recall To enhance the model's capabilities in specialized areas, it is essential to include ample knowledge documents related to mathematics, code, and reasoning. For this purpose, we extract relevant documents from unused web pages by training `fasttext` (Bojanowski et al., 2017) and TinyBert (Jiao et al., 2020) classifiers specifically tailored to these categories. From the FineWeb-Edu (Lozhkov et al., 2024a) and DCLM (Li et al., 2024b) web corpus, we extract 10.4B math text tokens, 1.11B code text tokens, and 1.01B reasoning text tokens. which are directly used for training or serve as seed data for synthesizing instruction data. Furthermore, we reuse the synthesized science data (1.5B) from Llama-3-SynE (Chen et al., 2024), which covers an extensive range of disciplines, such as math and physics.

Model-based quality scoring For general web page data and mathematical pre-training data, we use the `fineweb-edu-scorer` released by FineWeb-Edu for data scoring. For Python code data, we use the `python-edu-scorer` released by FineWeb-Edu. To avoid language models favoring highly technical pages like arXiv abstracts and submitted papers, these two classifiers focus on knowledge at the elementary and middle school levels. Following the methodology of (Penedo et al., 2024), we conduct quality assessments on all Python code data, most mathematical data, and web page data using scoring tools. We exclude data with scores of 1 and 2 and then heuristically sort data with scores from 3 to 5 .

Decontamination To ensure the fairness of comparison, we perform decontamination based on the selected evaluation benchmarks. Initially, we tokenize both the training set and the benchmarks that require decontamination, such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), HumanEval (Chen et al., 2021), and ARC (Yadav et al., 2019). Next, we divide all the benchmarks using n-gram tokens to create a contamination set. We use tokens rather than words to form n-gram segment, which achieves a higher level of decontamination in the domains of mathematics and code. Additionally, we exclude 20-gram segments that occur more than four times, as they are typically not relevant to the questions or solutions. Ultimately, the contamination set comprises 1,917,428 tuples. For each training document, if more than 10% of its generated 20-grams are present in the contamination set, we exclude that document from the final pre-training set.

E Post-training Details

We conduct post-training for YuLan-Mini, with specific details for each stage as described below. Experimental results of post-training on public benchmarks are shown in Table 3.

E.1 SFT Stage

During the Supervised Fine-Tuning (SFT) phase, we implement comprehensive optimization of training data through the following core strategies:

Diversified Data Sources Our SFT data comprises two categories: 1) high-quality open-source general-purpose data spanning diverse domains and topics, and 2) specialized data generated through synthesis, distillation, and paraphrasing techniques to ensure broad knowledge coverage and strong domain adaptability.

Rigorous Data Filtering Beyond conventional deduplication and filtering, our pipeline incorporates multi-stage quality control measures, including corpus quality assessment and curriculum learning-based selection to optimize training effectiveness.

Systematic Data Schedule We strategically balance proportions between general-purpose and specialized data based on their respective characteristics. Furthermore, we dynamically adjust data ratios according to real-time training feedback to achieve better performance.

E.2 DPO Stage

In the Direct Preference Optimization (DPO) phase, we adopt a hybrid data sampling strategy: 1) sampling from the SFT instruction dataset, and 2) incorporating diverse external instructions. Responses are generated using both our SFT-tuned model and high-performing open-source models. To ensure response quality, we utilize open-source models for evaluation and filtering, ultimately constructing high-quality preference datasets containing both on-policy and off-policy samples. This DPO training significantly enhances the model's capabilities in mathematical reasoning, code generation, and instruction adherence.

E.3 PPO Stage

Building upon the DPO-enhanced model, we employ Proximal Policy Optimization (PPO) with a dual-reward mechanism: combining RM-based rewards with rule-based rewards. The latter proves particularly effective in verifiable domains like mathematics and instruction following. Our training dataset comprises thousands of samples covering diverse task scenarios, which enables robust policy optimization.