A Learning Rate Path Switching Training Paradigm for Version Updates of Large Language Models

Anonymous ACL submission

Abstract

Due to the continuous emergence of new data, version updates have become an indispensable requirement for Large Language Models (LLMs). The training paradigms for version updates of LLMs include pre-training from scratch (PTFS) and continual pre-training (CPT). Preliminary experiments demonstrate that PTFS exhibits better pre-training performance, while the training cost of CPT is lower. Moreover, their performance and training cost gaps gradually widen with the version updates processing. To investigate the underlying reasons for this phenomenon, we analyze the effect of learning rate adjustments during the two stages of CPT: preparing an initialization checkpoint and conducting pre-training based on this checkpoint. We find that a large learning rate in the first stage and a complete learning rate decaying process in the second stage are crucial for version updates of LLMs. Hence, we propose a learning rate path switching training paradigm. Our paradigm comprises one main path, where we pre-train a LLM with the maximal learning rate, and multiple branching paths, each of which corresponds to an update of the LLM with newly-added training data. Compared with PTFS, when training four versions of LLMs, our paradigm can reduce the total training cost to 58% while maintaining comparable pre-training performance. In addition, we also validate the generalization of our paradigm, further proving its practicability.

1 Introduction

In recent years, there has been significant progress in the research of Large Language Models (LLMs). By performing large-scale training on massive datasets, LLMs have demonstrated remarkable capabilities, contributing to various fields (Wu et al., 2023; Cui et al., 2023; Wang et al., 2024; Guo et al., 2024). However, the training cost of LLMs is significantly higher than that of traditional NLP models. Particularly, in practical applications, LLMs have to face the need for *version updates* due to the continuous emergence of new data, which exacerbates the training cost of LLMs. Therefore, how to reduce the training cost while ensuring optimal pre-training performance across different versions has become one of the pivotal challenges of LLMs.

Generally, training paradigms applicable for updating LLMs can be categorized into two types: 1) Pre-Training From Scratch (PTFS): retraining new versions of LLMs on both old and new data. The well-known LLMs including LLaMA (Touvron et al., 2023a,b), GLM (Zeng et al., 2023), and Baichuan (Yang et al., 2023) are updated via this paradigm. 2) Continual Pre-Training (CPT): further pre-training new versions of LLMs on only new data based on the checkpoints from old versions. This paradigm is often utilized in resource constrained scenarios, such as limited computational resources or unavailability of old data.

In this paper, we firstly conduct preliminary experiments to compare the above two paradigms in version updates of LLMs. Compared with PTFS, CPT uses previous checkpoints for initialization, resulting in lower total training cost. However, CPT suffers from the inferior pre-training performance, which becomes increasingly serious as version updates processing. To study the reasons for this phenomenon, we break down the CPT process into two stages: the first stage for preparing an initialization checkpoint, and the second stage for continual pre-training based on the initialization checkpoint. Then, we conduct two groups of experiments to analyze the effect of learning rate adjustments during these two stages, obtaining two conclusions: 1) the larger the learning rate in the first stage, the better the performance of updated LLMs in the second stage; 2) for the second stage, a complete learning rate decaying process is beneficial to ensure the optimal performance of updated LLMs.

Based on the above analyses, we propose a learning rate path switching training paradigm for ver-

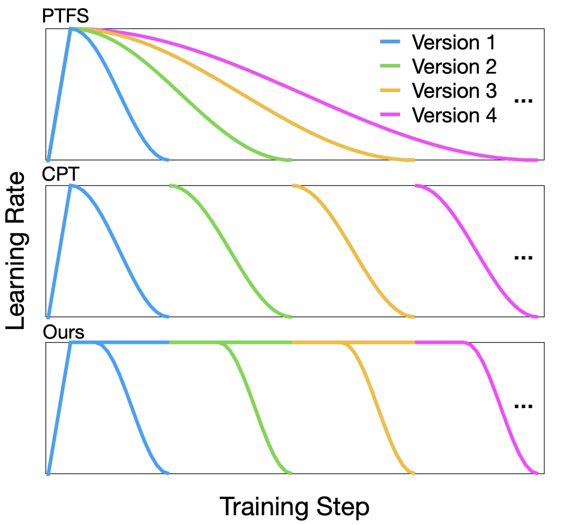

Figure 1: Learning rate curves of cosine learning rate schedule under PTFS, CPT[1] and our paradigm, all of which are used to update four versions of LLMs. Here, different color curves represent different version updates of LLMs.

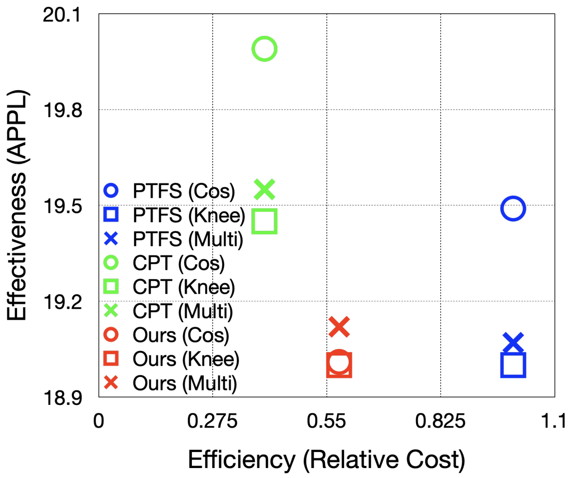

Figure 2: Comparison of different training paradigms. "APPL" (↓) denotes the average perplexity of LLMs across different versions, "Relative Cost" (↓) is the ratio of the total training steps across different versions of each paradigm to the total training steps of PTFS. The lower left corner achieves the best trade-off.

sion updates of LLMs. To better illustrate our paradigm, we take the most commonly used cosine learning rate schedule (Smith and Topin, 2019) as an example, and plot the learning rate curves of PTFS, CPT and our paradigm in Figure 1. Please note that our paradigm is also applicable to other schedules, such as Knee (Iyer et al., 2023), and multi-step (Bi et al., 2024) learning rate schedules.

In short, the learning rate curve of our paradigm, comprises one main path and multiple branching paths, each of which corresponds to a version update of LLM. As shown by the main path of Figure 1, we pre-train a LLM with the maximal learning rate, providing superior initialization checkpoints for subsequent continual pre-training. When we want to update the LLM with newly-added training data, we perform continual pre-training on the LLM with a dynamically-adjusted learning rate. Back to Figure 1, after a few steps of training with the maximal learning rate, the learning rate fast decays to its minimum, which effectively ensures the training performance of the updated LLM. Meanwhile, on the main path, we continue to pre-train the original checkpoint with the maximal learning rate, facilitating subsequent LLM updates.

Our paradigm better balances model performance and training cost compared to the other two paradigms, as detailed in Figure 2. To summarize, our main contributions are as follows:

- We conduct preliminary experiments to compare PTFS and CPT in the version updates of LLMs. Furthermore, our in-depth analyses show that using a large learning rate at the beginning and subsequent learning rate decay are crucial for improving the performance of updated LLMs.
- We propose a learning rate path switching paradigm for version updates of LLMs. To the best of our knowledge, our work is the first attempt to explore how to balance model performance and training cost for version updates of LLMs.
- Experimental results and in-depth analyses strongly demonstrate the effectiveness and generalization of our paradigm. Particularly, when training four versions of LLMs, our paradigm is able to achieve comparable pre-training performance to PTFS with only 58% of total training cost.

2 Preliminary Study

In this section, we first compare the performance of PTFS and CPT in version updates of LLMs, and then analyze the underlying reasons for their performance gap.

[1]In fact, multiple CPT variants can be used to version updates of LLMs. We compare these variants in Appendix B, and only retain the best performing variant in the subsequent experiments.

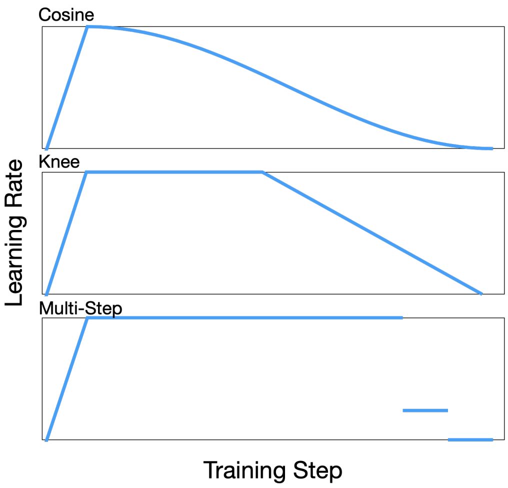

Figure 3: Learning rate curves of cosine (Smith and Topin, 2019), Knee (Iyer et al., 2023), and multi-step (Bi et al., 2024) learning rate schedules.

LRS	TP	Cost	PPL		
			V2	V3	V4
Cos	PTFS	1.00×	20.84	19.28	18.36
	CPT	0.40×	21.11	19.70	18.87
	Δ	-	-0.27	-0.42	-0.51
Knee	PTFS	1.00×	20.22	18.80	17.98
	CPT	0.40×	20.56	19.27	18.52
	Δ	-	-0.34	-0.47	-0.54
Multi	PTFS	1.00×	20.28	18.88	18.06
	CPT	0.40×	20.62	19.37	18.65
	Δ	-	-0.34	-0.49	-0.59

Table 1: Comparison between PTFS and CPT for training four versions of LLMs. "LRS" and "TP" indicate learning rate schedule and training paradigm, respectively. "V*" means the *-th version of LLM. Notably, whether using PTFS or CPT, the learning rate curve and pre-training performance of the first version remain unchanged. Thus, we do not report the performance of the first version in all experiments.

2.1 Setup

Model In this study, we use LLaMA-1.2B (Touvron et al., 2023a,b) as our base LLM and train for four versions. When employing PTFS, the total training steps for these four versions are 10K, 20K, 30K, and 40K, respectively. For CPT, each LLM update only requires 10K training steps. We train all LLMs with a batch size of 1.05M tokens.

Learning Rate Schedule We conduct experiments with three learning rate schedules: cosine (Smith and Topin, 2019), Knee (Iyer et al., 2023), and multi-step (Bi et al., 2024) learning rate schedules.[2] The specific learning rate curves of these schedules are plotted in Figure 3. Notably, cosine learning rate schedule is the most commonly used one for training LLMs (Zhao et al., 2023), and both Knee and multi-step learning rate schedules can achieve comparable or even superior performance than cosine learning rate schedule. For all learning rate schedules, we implement a linear warm-up phase of 2K steps (approximately 2.1B tokens). Besides, we set the maximum and minimum learning rates for these schedules to 3e-4 and 3e-5, respectively.

Dataset Similar to LLaMA (Touvron et al., 2023a,b), our training corpus comprises a mixture of data from publicly available sources, including code, paper, Wikipedia, books, mathematics, CommonCrawl and C4, webpage, translation and others. In total, our training data contains 764M English and Chinese samples. Due to the limitation of GPU resource, we do not experiment with the entire dataset. To simulate the scenario of version updates, we perform non-replacement sampling on the training data to obtain 10.5B tokens as the newly-added data for each update. Hence, when using PTFS, we train four versions of LLMs from scratch with 10.5B, 21B, 31.5B, and 42B tokens, respectively. By contrast, using CPT to update the LLMs only involves the newly-added 10.5B tokens each time.

Evaluation Following previous studies (Qin et al., 2022; Gupta et al., 2023; Bi et al., 2024), we mainly use perplexity (PPL) to evaluate the pre-training performance of LLMs. Meanwhile, we also focus on the training cost of each paradigm, defined as the total training steps required for different versions.

2.2 Comparison between PTFS and CPT

Experimental results are shown in Table 1. It is evident that CPT incurs a lower training cost, whereas PTFS achieves superior performance. More importantly, as the version updates processing, the performance gap between PTFS and CPT progressively widens.

To explore the underlying reasons for the above phenomenon, we still use cosine learning rate

[2]We also evaluate constant and inverse square root learning rate schedules, both of which perform worse than the three selected schedules.

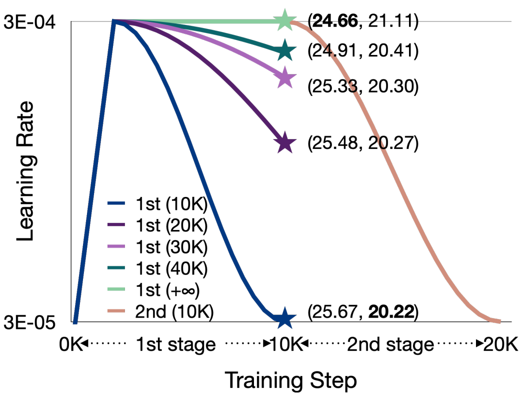

Figure 4: The effect of learning rate adjustment in the first stage. In the first stage, we vary the cosine cycle length as 10K, 20K, 30K, 40K and $+\infty$ steps, respectively, where the checkpoints at the 10K-th steps are selected as the initialization ones for the subsequent 10K-steps continual pre-training. "$(\cdot,\cdot)$" indicates the PPLs of the initialization checkpoint and corresponding updated LLM.

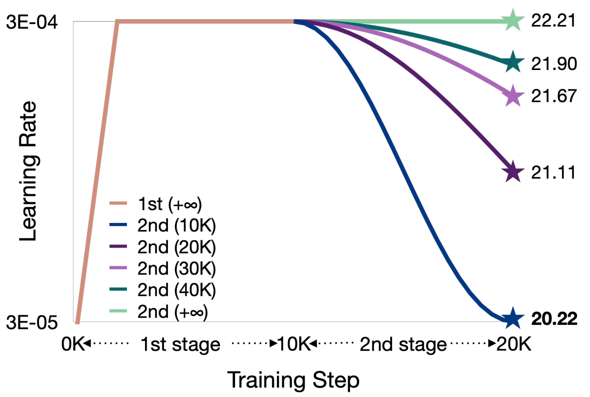

Figure 5: The effect of learning rate adjustment in the second stage. In the first stage, we directly use the maximal learning rate after warm-up. During the second stage, we try cosine cycle length with 10K, 20K, 30K, 40K and $+\infty$ steps, respectively, where the PPLs of LLMs at the 20K-th steps are compared.

schedule to conduct two groups of experiments, so as to investigate the impact of learning rate on the performance of updated LLMs during the two stages of CPT: 1) preparing an initialization checkpoint, and 2) performing continual pre-training based on the prepared initialization checkpoint.

Effect of Learning Rate Adjustment During the First Stage As depicted in Figure 4, in the first group of experiments, we vary the cosine cycle length to 10K, 20K, 30K, 40K, and $+\infty$ steps, respectively. The checkpoints at the 10K-th steps are selected as initialization checkpoints for the second stage. Then, we continually pre-train LLMs for 10K steps, where the learning rate gradually decays from its maximum to minimum. Back to Figure 4, we can observe that with the increase of the cosine cycle length in the first stage, the performance of an initialization checkpoint drops, whereas its corresponding updated LLM performs better. Therefore, we conclude that large learning rate in the first stage is advantageous for continual pre-training during the second stage.

Effect of Learning Rate Adjustment During the Second Stage Based on the above conclusion, we directly set the cosine cycle length in the first stage as $+\infty$ steps, as illustrated in Figure 5. Then, during continual pre-training, we experiment with the cosine learning rate schedule using different cosine cycle lengths: 10K, 20K, 30K, 40K, $+\infty$ steps, and report the performance of updated LLMs at the 20K-th steps. As shown in Figure 5, it is evident that **the complete decay of learning rate enables the updated LLMs to achieve the best performance**. This finding is consistent with the results from the first group of experiments mentioned above. In other words, when the learning rate experiences complete decay during the first stage, the initialization checkpoint's performance is also optimal.

Based on the findings of the above two groups of experiments, we can conclude that it is difficult for CPT to achieve good performance across different versions of LLMs. Specifically, according to the findings of the second group of experiments, if the current LLM is expected to achieve the best performance, its learning rate at the second stage should undergo a complete learning rate decay. However, such decay will result in a lower learning rate in the first stage of the next update of LLM, further degrading the performance of the updated LLM.

3 Our Paradigm

Based on the conclusions from Section 2, we propose a learning rate path switching paradigm for version updates of LLMs in this section. Our training cost is lower than PTFS and we achieve significantly better performance than CPT, even comparable to PTFS.

3.1 Paradigm Overview

Let us revisit Figure 1, which shows the learning rate curves of our paradigm applied to cosine learning rate schedule. Please note that our paradigm

LRS	α	Cost	PPL		
			V2	V3	V4
Cos	0.2	**0.49**×	20.34	19.13	18.44
	0.4	0.53×	20.16	18.91	18.21
	0.6	0.58×	**20.13**	18.81	18.09
	0.8	0.62×	20.15	**18.77**	**18.02**
Knee	0.2	**0.49**×	20.33	19.12	18.42
	0.4	0.53×	20.16	18.91	18.20
	0.6	0.58×	**20.12**	18.81	18.08
	0.8	0.62×	20.15	**18.77**	**18.01**
Multi	0.2	**0.49**×	20.33	19.08	18.37
	0.4	0.53×	**20.29**	18.91	18.16
	0.6	0.58×	20.40	**18.88**	18.09
	0.8	0.62×	20.63	18.91	**18.06**

Table 2: The effect of hyper-parameter α on the pre-training performance and training cost of our paradigm. Experiments are conducted on LLaMA-1.2B.

is also applicable to other schedules, such as Knee and multi-step and so on. Without loss of generality, the learning rate curve of our paradigm comprises one main path and multiple branching paths, each of which corresponds to one version update. On the main path, we pre-train the LLM from scratch with the maximal learning rate, providing initialization checkpoints for subsequent version updates. When we want to obtain an updated LLM, we directly use the current checkpoint of the main path as the initialization one, and then perform continual pre-training. During this process, the learning rate undergoes a complete fast-decaying process, effectively ensuring the performance of the updated LLM. Meanwhile, on the main path, we still use newly-added data to pre-train the existing checkpoint with the maximal learning rate, so as to facilitate subsequent updates.

Obviously, our paradigm has lower training cost than PTFS, as it conducts continual pre-training based on the initialization checkpoints from the main path. Unlike CPT, these checkpoints are obtained through training from scratch with the maximum learning rate, which enables the updated LLMs to achieve better performance, as analyzed in Section 2. The following experiments also fully confirm the superiority of our paradigm in balancing model performance and training cost.

LRS	TP	Cost	PPL		
			V2	V3	V4
Cos	PTFS	1.00×	20.84	19.28	18.36
	CPT	**0.40**×	21.11	19.70	18.87
	Ours	0.58×	**20.13**	**18.81**	**18.09**
Knee	PTFS	1.00×	20.22	**18.80**	**17.98**
	CPT	**0.40**×	20.56	19.27	18.52
	Ours	0.58×	**20.12**	18.81	18.08
Multi	PTFS	1.00×	**20.28**	**18.88**	**18.06**
	CPT	**0.40**×	20.62	19.37	18.65
	Ours	0.58×	20.40	**18.88**	18.09

Table 3: Comparison of different paradigms for training LLaMA-1.2B of different versions.

3.2 Time Complexity Analysis

To further compare different training paradigms in training cost, we define their time complexity functions as the total training steps of version updates.

Before providing our definitions, we first introduce two symbols to facilitate the subsequent descriptions: 1) N_v: the number of version updates os LLMs; 2) T: assuming a consistent number of data is added for each update. When updating the $i-th$ version of LLMs, PTFS requires updating $iT(1 \leq i \leq N_v)$ steps each time, CPT needs to train for T steps, and our paradigm requires training $T + \alpha T$ steps, where α $(0 \leq \alpha \leq 1)$ controls the proportion of fast-decaying steps to the total steps in each update.

Formally, the total training cost for each paradigm can be described as follows:

$$\begin{aligned}
\mathbf{C}_{\text{ptfs}}(N_v) &= \sum_{i=1}^{N_v} iT = 0.5TN_v^2 + 0.5TN_v, \\
\mathbf{C}_{\text{cpt}}(N_v) &= \sum_{i=1}^{N_v} T = TN_v, \\
\mathbf{C}_{\text{ours}}(N_v) &= \sum_{i=1}^{N_v-1} (T + \alpha T) + T \\
&= (1+\alpha)TN_v - \alpha T.
\end{aligned}$$

Please note that, for the last version, the addtional main path training for preparing initialization checkpoint for the next update can be omitted, which counts for αT steps, so only T steps are required.

Comparing the above functions, we can clearly find that $\mathbf{C}_{\text{ptfs}}(N_v)$ is a quadratic function with re-

Ver.	TP	C^3	GSM8K	MMLU	CSL	C-EVAL	BBH	CMMLU	GAOKAO	AGIEval	AVG
V2	PTFS	38.00	4.63	**24.00**	38.25	**30.09**	17.43	25.37	18.10	14.59	23.38
	CPT	37.00	4.09	23.52	35.11	27.42	18.55	**25.63**	**18.86**	13.40	22.62
	Ours	**38.60**	**5.08**	22.94	**39.08**	28.38	**20.79**	24.88	18.48	**14.73**	**23.66**
V3	PTFS	40.30	3.34	**24.33**	**39.17**	25.85	17.11	**25.30**	**22.03**	14.34	23.53
	CPT	38.30	**4.70**	23.32	36.40	28.38	**21.11**	24.76	17.85	13.47	23.14
	Ours	**42.10**	4.63	23.22	34.91	**29.35**	19.70	24.73	19.24	**14.90**	**23.64**
V4	PTFS	35.70	4.25	**24.93**	38.75	27.04	16.73	**24.97**	**21.01**	14.10	23.05
	CPT	**43.90**	4.55	22.20	38.69	27.19	21.62	24.43	18.23	13.50	23.81
	Ours	41.90	**5.53**	24.09	**40.24**	**27.71**	**21.84**	24.78	17.24	**14.40**	**24.19**

Table 4: Performance of LLMs across different versions on downstream tasks. "Ver." indicates the number of version of LLMs. More experimental results of LLMs on larger model size or larger data size are listed in Table 11.

spect to N_v, while both $\mathbf{C}_{\text{cpt}}(N_v)$ and $\mathbf{C}_{\text{ours}}(N_v)$ are linear ones. Besides, the gaps of $\mathbf{C}_{\text{ptfs}}(N_v)$ compared to the other two functions significantly widen with the increase of N_v. For example, when $N_v = 4$, the values of these three time complexity functions are $10T$, $4T$, $5.8T$, respectively. In contrast, they will separately reach $55T$, $10T$, $15.4T$ with $N_v = 10$.

4 Experiment

In this section, we still use the settings of the preliminary study to conduct more experiments, comparing the performance and training cost of different training paradigms.

4.1 Effect of Hyper-Parameter α

As described in Section 3, α is the most important hyper-parameter in our paradigm, as it controls the proportion of fast decay steps to the total number of steps. The fast decay steps influence model performance and training cost of our paradigm. To select an optimal α value, we try different α for our paradigm: from 0.2 to 0.8 with an increment of 0.2 each time, and then observe the changes in the pre-training performance and training cost of our paradigm.

Experimental results are listed in Table 2, which show that the overall performance of LLMs across different versions is optimal when α is set as 0.6 and 0.8. However, when α is set to 0.6, our paradigm achieves lower training cost. Thus, we directly use $\alpha = 0.6$ in subsequent experiments.

4.2 Main Experiments

Then, we compare different paradigms in terms of pre-training performance and downstream performance. To comprehensively examine our paradigm, we conduct experiments with the previously-mentioned three learning rate schedules.

Pre-Training Performance From Table 3, we can observe that compared with PTFS, **our paradigm reduces the total training cost to 58% while maintaining comparable pre-training performance**. Particularly, when using cosine learning rate schedule, our paradigm can even slightly outperforms PTFS. On the other hand, as expected, the training cost of our paradigm is still higher than that of CPT, however, it always achieves better performance than CPT, no matter which schedule is used. Overall, our paradigm achieves a better balance between pre-training performance and total training cost of LLMs during version updates of LLMs.

Performance on Downstream Tasks Furthermore, we investigate the performance of different training paradigms on benchmarks of nine downstream tasks, including C^3 (Sun et al., 2020), GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), CSL (Li et al., 2022), C-EVAL (Huang et al., 2023), BBH (Suzgun et al., 2023), CMMLU (Li et al., 2023), GAOKAO (Zhang et al., 2023), AGIEval (Zhong et al., 2023). To this end, we first construct a general SFT dataset with 1.8B tokens and then perform SFT on each of the four versions of updated LLMs.

From the results listed in Table 4, we can clearly find that our paradigm can still obtain better average performance than both PTFS and CPT, further proving the effectiveness of our paradigm.

LRS	TP	Cost	PPL		
			V2	V3	V4
Cos	PTFS	1.00×	20.94	19.35	18.41
	CPT	**0.40×**	21.23	19.78	18.92
	Ours	0.58×	**20.23**	**18.87**	**18.11**
Knee	PTFS	1.00×	20.30	**18.84**	**17.98**
	CPT	**0.40×**	20.67	19.34	18.56
	Ours	0.58×	**20.20**	18.85	18.09
Multi	PTFS	1.00×	**20.37**	**18.92**	**18.06**
	CPT	**0.40×**	20.74	19.44	18.68
	Ours	0.58×	20.49	**18.92**	18.09

Table 5: The generalization of our paradigm in terms of model architecture. Based on Qwen-1.2B, we conduct experiments with the same setting as LLaMA-1.2B.

Size	TP	PPL		
		V2	V3	V4
203M	PTFS	30.97	29.50	28.65
	CPT	31.31	29.90	29.07
	Ours	**30.25**	**28.94**	**28.19**
406M	PTFS	26.58	25.06	24.19
	CPT	26.89	25.49	24.67
	Ours	**25.85**	**24.52**	**23.79**
608M	PTFS	23.12	21.75	20.93
	CPT	23.50	22.26	21.52
	Ours	**22.59**	**21.43**	**20.77**
1.2B	PTFS	20.84	19.28	18.36
	CPT	21.22	19.79	18.97
	Ours	**20.13**	**18.81**	**18.09**
2.1B	PTFS	18.33	16.88	16.04
	CPT	18.76	17.47	16.72
	Ours	**17.82**	**16.63**	**15.97**
3.1B	PTFS	17.22	15.87	**15.07**
	CPT	17.67	16.48	15.77
	Ours	**16.84**	**15.72**	15.09

Table 6: The generalization of our paradigm in terms of model size. The model sizes range from 203M to 3.1B.

4.3 Generalization of Our Paradigm

Subsequently, we explore the generalization of our paradigm in the following aspects: model architecture, model size, data size, and maximum learning rate, all of which are crucial for practical applications of LLMs. During this process, we still use cosine learning rate schedule.

Model Architecture To demonstrate the generalization of our paradigm in model architecture, we use Qwen-1.2B (Bai et al., 2023) to re-conduct experiments with the same setting as LLaMA-1.2B.

We list the experiments results in Table 5, which indicate that our paradigm is also applicable to other model architecture.

Model Size We then focus on the generalization of our paradigm on model scaling. To this end, we vary the number of model parameters to conduct experiments. In total, we consider six model sizes: 203M, 406M, 608M, 1.2B, 2.1B, 3.1B, of which detailed hyper-parameters are listed in Appendix A.

From the results shown in Table 6, we observe that our paradigm achieves pre-training performance comparable to PTFS across different sizes of LLMs and outperforms CPT.

Data Size Next, we switch our attention to the applicability of our paradigm in data size. To do this, we re-conduct experiments using different sizes of total training data: 21B, 42B, and 168B. Correspondingly, the training steps are 5K, 10K and 40K for each LLM update, respectively.

As shown in the experimental results in Table 7, our paradigm achieves optimal pre-training performance across different data sizes, which further demonstrates the generalization of our paradigm.

Maximum Learning Rate Finally, we aim to verify the generalization of our paradigm with respect to the maximum learning rate. We conduct experiments by setting the maximum learning rates as 5e-5, 1e-4, 3e-4, 5e-4, 8e-4, respectively.

From Table 8, as the maximum learning rate increases, our paradigm always achieves better or comparable performance than PTFS, let alone CPT. This strongly highlights the generalization of our paradigm in the maximum learning rate.

5 Related Work

Continual Training Continual training is one of the most direct approaches for version updates of LLMs. Related studies of continual training can be broadly categorized into the following three types: 1) Methods introducing additional parameters (Ke et al., 2022, 2023; Song et al., 2023; PENG et al., 2024); 2) Prompt-based methods (Wang et al., 2022b,a; Razdaibiedina et al., 2023); 3) Scenario-specific methods (Peng et al., 2023; Gogoulou et al.,

Data	TP	PPL		
		V2	V3	V4
21B	PTFS	24.66	22.31	20.84
	CPT	25.10	22.84	21.56
	Ours	**23.59**	**21.41**	**20.27**
42B	PTFS	20.84	19.28	18.36
	CPT	21.11	19.70	18.87
	Ours	**20.13**	**18.81**	**18.09**
168B	PTFS	16.70	15.97	15.54
	CPT	16.90	16.25	15.86
	Ours	**16.47**	**15.86**	**15.51**

Table 7: The generalization of our paradigm in terms of data size. The total data sizes (for four versions) range from 21B to 168B.

MLR	TP	PPL		
		V2	V3	V4
5e-5	PTFS	34.78	29.53	26.65
	CPT	35.23	30.08	27.23
	Ours	**29.99**	**25.54**	**23.27**
1e-4	PTFS	26.34	23.28	21.57
	CPT	26.64	23.70	22.04
	Ours	**23.89**	**21.32**	**19.97**
3e-4	PTFS	20.84	19.28	18.36
	CPT	21.22	19.79	18.97
	Ours	**20.13**	**18.81**	**18.09**
5e-4	PTFS	19.89	18.62	**17.85**
	CPT	20.17	19.05	18.38
	Ours	**19.53**	**18.45**	**17.85**
8e-4	PTFS	19.38	**18.26**	**17.58**
	CPT	19.69	18.73	18.16
	Ours	**19.22**	18.30	17.78

Table 8: The generalization of our paradigm in terms of the maximum learning rate. "MLR" indicates the maximum learning rate.

2023a; Xie et al., 2023). Significantly different from the above studies, our paradigm comprises one main learning rate path, where we perform pre-training with the maximal learning rate, and multiple learning rate branches with the complete decay process. Thus, our paradigm achieve a better trade-off between the performance and cost.

Learning Rate The learning rate is one of the most crucial hyper-parameters for training LLMs. Existing learning rate schedules can be broadly divided into the following four categories according to their policies (Wu et al., 2019; Wu and Liu, 2023; Jin et al., 2023): 1) Fixed learning rate policy, such as constant learning rate schedule; 2) Decaying learning rate policy, such as inverse square root learning rate schedule; 3) Cyclic learning rate policy, such as cosine learning rate schedule; 4) Composite learning rate policy, such as Knee and multi-step learning rate schedules. In addition, there are some recent studies exploring learning rate schedules for LLMs, including Warmup-Stable-Decay schedule (Hu et al., 2024) and Constant Learning Rate with Cooldown (Hägele et al., 2024). Particularly, our paradigm is a well-designed training paradigm for version updates of LLMs, which is applicable to cosine, Knee, and multi-step and other learning rate schedules.

6 Conclusion and Future Work

In this paper, we mainly focus on how to better balance model performance and training cost for version updates of LLMs. Through the analysis in the preliminary study, we find that 1) a large learning rate is beneficial for providing a better initialization checkpoints for subsequent updates, and 2) a complete learning rate decay process enables the updated LLMs to achieve optimal performance. Based on the above two findings, we propose a learning rate path switching paradigm for version updates of LLMs, which comprises one main path and multiple branching paths. On the main path, we pre-train the LLMs with the maximal learning rate to provide superior initialization checkpoints for subsequent updates. Each time an update is required, our paradigm switches from the main path to a branching path, undergoing a complete learning rate decay process. Experimental results and further analyses strongly demonstrate the effectiveness and generalization of our paradigm.

In the future, we will further expand the practical scope of our paradigm. Current research mainly focuses on the pre-training phase and does not include supervised fine-tuning, safety alignment, etc., which can be incorporated into the fast decay phase of our paradigm. Additionally, we plan to explore the applicability of our paradigm based on multimodal large models.

Limitations

Although the training cost of our paradigm is significantly lower than that of PTFS, it is still higher than that of CPT. Hence, we plan to design a precise method to determine the proportion of the fast-decaying steps to the total steps, which can further reduce the training cost of our paradigm.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, et al. 2023. Qwen technical report. arxiv.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. arXiv.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, et al. 2021. Training verifiers to solve math word problems. arXiv.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, et al. 2023. A survey on multimodal large language models for autonomous driving. arXiv.

Evangelia Gogoulou, Timothée Lesort, Magnus Boman, and Joakim Nivre. 2023a. A study of continual learning under language shift. arXiv.

Evangelia Gogoulou, Timothée Lesort, Magnus Boman, and Joakim Nivre. 2023b. A study of continual learning under language shift. arXiv.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, et al. 2024. Large language model based multi-agents: A survey of progress and challenges. arXiv.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, et al. 2023. Continual pre-training of large language models: How to (re)warm your model? In *ICML Workshop*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. arxiv.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, et al. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *NeurIPS*.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling laws and compute-optimal training beyond fixed training durations. arxiv.

Nikhil Iyer, V Thejas, Nipun Kwatra, Ramachandran Ramjee, and Muthian Sivathanu. 2023. Wide-minima density hypothesis and the explore-exploit learning rate schedule. *JMLR*.

Hongpeng Jin, Wenqi Wei, Xuyu Wang, Wenbin Zhang, Hongpeng Wu, YanzhaoJin, Wenqi Wei, et al. 2023. Rethinking learning rate tuning in the era of large language models. In *CogMI*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, et al. 2020. Scaling laws for neural language models. arXiv.

Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. 2022. Continual training of language models for few-shot learning. In *EMNLP*.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. In *ICLR*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, et al. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. arXiv.

Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. 2022. CSL: A large-scale Chinese scientific literature dataset. In *COLING*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Bohao PENG, Zhuotao Tian, Shu Liu, Ming-Chang Yang, and Jiaya Jia. 2024. Scalable language model with generalized continual learning. In *ICLR*.

Guangyue Peng, Tao Ge, Si-Qing Chen, Furu Wei, and Houfeng Wang. 2023. Semiparametric language models are scalable continual learners. arXiv.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. ELLE: Efficient lifelong pre-training for emerging data. In *Findings of ACL*.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *ICLR*.

Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-domain Operations Applications*.

Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, et al. 2023. Conpet: Continual parameter-efficient tuning for large language models. arXiv.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging Chinese machine reading comprehension. In *TACL*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of ACL*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, et al. 2023a. Llama: Open and efficient foundation language models. arXiv.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. arXiv.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, et al. 2024. A survey on large language model based autonomous agents. *Frontiers Comput*.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, et al. 2022a. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, et al. 2022b. Learning to prompt for continual learning. In *CVPR*.

Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S. Yu. 2023. Multimodal large language models: A survey. In *IEEE BigData*.

Yanzhao Wu and Ling Liu. 2023. Selecting and composing learning rate policies for deep neural networks. *ACM TIST*.

Yanzhao Wu, Ling Liu, Juhyun Bae, Ka-Ho Chow, Arun Iyengar, Calton Pu, et al. 2019. Demystifying learning rate policies for high accuracy training of deep neural networks. In *IEEE BigData*.

Yong Xie, Karan Aggarwal, and Aitzaz Ahmad. 2023. Efficient continual pre-training for building domain specific large language models. arXiv.

Aiyuan Yang, Bin Xiao, Bingning Xiao, Borong Zhang, Ce Bian, Chao Yin, et al. 2023. Baichuan 2: Open large-scale language models. arXiv.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, et al. 2023. Glm-130b: An open bilingual pre-trained model. In *ICLR*.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. arXiv.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, et al. 2023. A survey of large language models. arXiv.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. arXiv.

Size	MLR	Hidden	Head	Layer
203M	1e-3	512	8	24
406M	6e-4	1,024	16	12
608M	6e-4	1,024	16	24
1.2B	3e-4	1,536	16	24
2.1B	3e-4	1,536	16	48
3.1B	3e-4	8,192	32	40

Table 9: Detailed hyper-parameters of LLMs with different sizes.

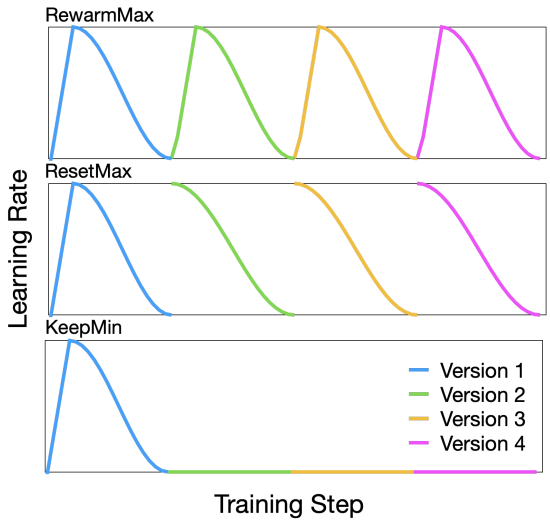

Figure 6: Learning rate curves of different adaptation method of CPT for version iteration of LLMs. The learning rate curves are plotted based on cosine learning rate schedules.

A Detailed Hyper-Parameters

In this work, we compare PTFS, CPT and our paradigm based on LLMs with different sizes, of which hyper-parameters are listed in Table 9. Following Kaplan et al.; Mann et al., we set smaller maximum learning rates for larger LLMs. Besides, we set the minimum learning rate as 0.1 times of the maximum learning rate for all LLMs.

B CPT Variants

In order to adapt traditional CPT for version updates of LLMs, we compare three variants of CPT in Figure 6:

- **RewarmMax**: Warm up the leaning rate periodically, and use the learning rate schedule of old version to train the new version of LLMs (Gupta et al., 2023).

LRS	Variant	PPL		
		V2	V3	V4
Cos	RewarmMax	21.22	19.79	18.97
	ResetMax	**21.11**	**19.70**	**18.87**
	KeepMin	23.00	21.99	21.26
Knee	RewarmMax	20.74	19.46	18.70
	ResetMax	**20.56**	**19.27**	**18.52**
	KeepMin	22.22	21.36	20.37
Multi	RewarmMax	20.80	19.55	18.82
	ResetMax	**20.62**	**19.37**	**18.65**
	KeepMin	22.11	21.24	20.60

Table 10: Comparison among RewarmMax, ResetMax and KeepMin for CPT.

- **ResetMax**: Directly set the leaning rate as maximum periodically, and use the learning rate schedule of old version to train the new version of LLMs (Gupta et al., 2023).
- **KeepMin**: Keep the learning rate in minimum by using a constant learning rate schedules, so as to ensure the training convergence of LLMs (Gogoulou et al., 2023b).

Experimental results are listed in Table10. We can observe that ResetMax achieves the best pre-training performance among these variants. Therefore, we use ResetMax for the other experiments.

C Performance of Downstream Tasks

In addition to the standard training scale (LLaMA-1.2B trained for 42B tokens), we also evaluate LLMs with more training data (LLaMA-1.2B trained for 168B tokens) and larger model size (LLaMA-3.1B trained for 42B tokens). We report the performance of downstream tasks across different versions in Table 11. Experimental results show that our paradigm achieves superior average performance compared to PTFS and CPT across different training scales on downstream tasks.

Scale	Ver.	TP	C^3	GSM8K	MMLU	CSL	C-EVAL	BBH	CMMLU	GAOKAO	AGIEval	AVG
1.2B 42B	V2	PTFS	38.00	4.63	**24.00**	38.25	**30.09**	17.43	25.37	18.10	14.59	23.38
		CPT	37.00	4.09	23.52	35.11	27.42	18.55	**25.63**	**18.86**	13.40	22.62
		Ours	**38.60**	**5.08**	22.94	**39.08**	28.38	**20.79**	24.88	18.48	**14.73**	**23.66**
	V3	PTFS	40.30	3.34	**24.33**	**39.17**	25.85	17.11	**25.30**	**22.03**	14.34	23.53
		CPT	38.30	**4.70**	23.32	36.40	28.38	**21.11**	24.76	17.85	13.47	23.14
		Ours	**42.10**	4.63	23.22	34.91	**29.35**	19.70	24.73	19.24	**14.90**	**23.64**
	V4	PTFS	35.70	4.25	**24.93**	38.75	27.04	16.73	**24.97**	**21.01**	14.10	23.05
		CPT	**43.90**	4.55	22.20	38.69	27.19	21.62	24.43	18.23	13.50	23.81
		Ours	41.90	**5.53**	24.09	**40.24**	**27.71**	**21.84**	24.78	17.24	**14.40**	**24.19**
1.2B 168B	V2	PTFS	38.90	6.82	23.49	40.33	**29.27**	**23.28**	25.14	**23.29**	14.39	24.99
		CPT	**43.80**	7.13	24.61	37.22	26.52	22.96	25.40	20.13	14.25	24.67
		Ours	43.20	**8.95**	**25.43**	**40.45**	26.90	22.16	**25.45**	18.73	**15.94**	**25.25**
	V3	PTFS	47.40	8.49	25.04	42.42	**27.42**	**26.88**	**25.06**	18.23	16.59	**26.39**
		CPT	40.30	8.42	24.30	41.61	26.30	24.07	24.59	**20.00**	**18.00**	25.29
		Ours	**47.70**	**9.33**	**25.35**	**44.39**	25.85	23.05	24.85	17.60	15.63	25.97
	V4	PTFS	48.50	8.19	24.73	44.37	26.82	**25.70**	25.19	19.49	15.36	26.48
		CPT	**49.10**	8.34	25.48	40.60	**27.27**	22.54	25.38	21.39	**17.44**	26.39
		Ours	48.20	**9.02**	**26.30**	**44.56**	**27.27**	23.69	**25.56**	**22.53**	14.20	**26.81**
3.1B 42B	V2	PTFS	41.10	6.37	24.00	36.43	24.15	21.62	24.97	19.75	**14.22**	23.62
		CPT	**46.00**	6.14	24.00	**40.81**	**27.04**	21.94	23.57	**20.89**	13.28	24.85
		Ours	43.70	**8.57**	**24.23**	40.17	25.78	**24.59**	**25.70**	19.37	**14.22**	**25.15**
	V3	PTFS	44.30	8.34	23.83	40.99	**27.12**	21.71	24.73	**21.65**	**15.48**	25.35
		CPT	43.90	8.11	**25.23**	**41.24**	26.00	25.00	**25.44**	20.00	13.40	25.37
		Ours	**47.90**	**9.48**	24.02	40.74	25.71	**25.73**	25.09	19.62	14.54	**25.87**
	V4	PTFS	50.20	**11.22**	**25.98**	39.89	27.64	**23.12**	25.47	21.65	**15.46**	26.74
		CPT	**50.60**	9.78	25.12	41.03	**28.08**	22.48	25.38	21.01	13.93	26.38
		Ours	49.80	10.77	25.77	**42.95**	26.97	22.45	**26.25**	**22.41**	14.80	**26.91**

Table 11: Performance of downstream tasks of LLMs across four versions. In addition to the standard training scale (LLaMA-1.2B trained for 42B tokens), we also evaluate LLMs with more training data (LLaMA-1.2B trained for 168B tokens) and larger model size (LLaMA-3.1B trained for 42B tokens).