Enhancing LLM Pretraining by Checkpoint Merging: An Almost Free Lunch Approach.

Anonymous ACL submission

Abstract

Large language models (LLMs), such as GPT-4, LLaMA and Gemini, have achieved widespread success across a wide range of natural language processing (NLP) tasks. Pretraining is a foundational step in the LLM training process, where the model gains a general understanding of language by exposure to vast amounts of text data. However, pretraining LLM comes with high costs and significant impacts on energy consumption and the environment. To alleviate this issue, we propose a simple and almost free lunch approach, which involves merging the LLM's checkpoints that share training trajectories during the pretraining phase. Besides improving pretraining without increasing the compute budget, our method can relax the requirement of the label information in contrast to previous merging methods, which is achieved by leveraging generation quality as the indicator to determine the merging weight. Through various experiments, we demonstrate that the merged checkpoint can achieve superior performance across multiple datasets compared to the best-performing individual checkpoint and still exhibits higher generalization performance in the out-of-distribution setting.

1 Introduction

The field of NLP has recently undergone a revolution propelled by the emergence of large language models (such as Brown et al. (2020); Touvron et al. (2023); OpenAI (2023), *inter alia*). With the continuous growth in the scale of language models and training data, LLMs exhibit various emerging capabilities. It is capable of addressing diverse tasks by conditioning the models on just a few examples or task-descriptive instructions (Brown et al., 2020; Dong et al., 2023). This new paradigm has achieved impressive results in a range of tasks, including logical reasoning and common-sense inference (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022).

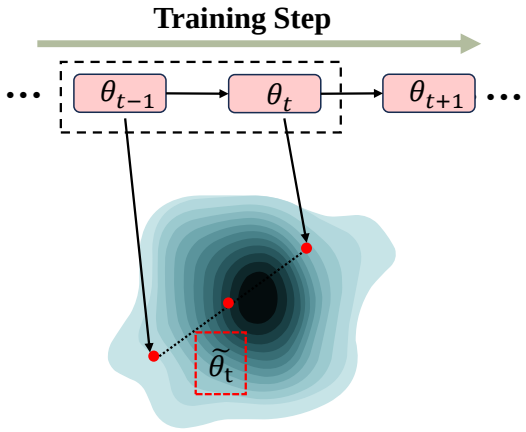

Figure 1: Illustration of Checkpoint Merging.

As we all know, training such a strong LLM from scratch incurs significant costs. For instance, training a Llama 2 70B model with 2T tokens needs 1,720,320 GPU hours (Touvron et al., 2023). Besides the substantial requirements of training data, advanced technology, computational resources and skilled programmers, training an LLM from scratch has a significant impact on energy consumption and the environment (Faiz et al., 2024). For instance, developing a transformer comprising 213 million parameters through neural architecture search has been likened to the carbon dioxide equivalent emissions of five cars over their entire lifespans (Strubell et al., 2019). Therefore, one crucial challenge within this domain is how to reduce consumption and cost during the pretraining phase.

Recent efforts on efficient LLM pretraining involve mixed-precision training (Shoeybi et al., 2020), pipeline parallelism (Liu et al., 2023), zero redundancy optimizer (Rajbhandari et al., 2020), depth up-scaling method (Kim et al., 2023), and so on. While these approaches contribute to efficient training with reduced computational cost, most of them focus on the model architecture or the optimization process (Hou et al., 2022).

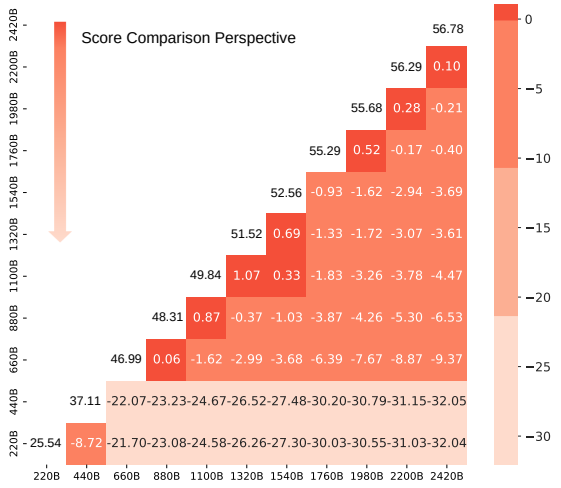

Figure 2: Performance Delta: 11 checkpoints Merged with Greedy Soup vs. Individual checkpoint Before Merging on CMMLU. The black numbers represent the original performance of the Checkpoint on two datasets.

Unlike recent studies on efficient LLM pretraining, we focus on a simple but efficient strategy to enhance pretraining with minimal computational expenditure, i.e., "model merging". Model merging is defined as combining multiple models with a common architecture into a single one (the result is referred to "soup") in parameter space, which can compensate for biases or errors that may exist in individual models in certain areas (Polyak and Juditsky, 1992; Wortsman et al., 2022). As a simple and efficient technology, model merging has attracted increasing attention in the study of LM. For example, Jin et al. (2023) study the problem of merging individual LM fine-tuned on different training data sets to obtain a single model that performs well both across all data set domains. Yu et al. (2024) focus on merging multiple homologous self-supervised fine-tuning LLMs to obtain new capabilities. Meanwhile, Wan et al. (2024) explore the merging of LLMs from a probabilistic distribution perspective for utilizing the collective capabilities and unique strengths of diverse LLMs.

While extensive research has been devoted to model merging in LLM, there remains a paucity of studies focused on employing the model merging strategy to mitigate consumption and costs during the pretraining phase. Besides, existing approaches, such as Wortsman et al. (2022); Matena and Raffel (2022), require the inclusion of a labeled dataset to determine the merging weights applicable to each model, but in practice, a labeled dataset usually incurs high annotation costs, e.g., law-related or medical-related questions that often require professional knowledge to answer (Fu et al., 2023). There are also some model merging methods without requiring labeled data, such as uniform soup (Wortsman et al., 2022), LAWA (Sanyal et al., 2023) and RegMean (Jin et al., 2023). However, they may result in low-precision models due to different local minima may be found in average weighted parameters (Utans, 1996; Chen et al., 2017).

To fill this gap, we make the following efforts in this paper: (1) Through pilot experiments, we initially investigate the characteristics of checkpoint merging and find that: (a) There is a higher probability of achieving performance enhancement when merging checkpoints that are adjacent during the pretraining phase; (b) Merging two checkpoints is wise, rather merging three or four checkpoints. Besides, we also find that there is a positive correlation between the generation quality and performance of LLM. (2) Based on these findings, we can impose a restriction on the merging of checkpoints and introduce a new merging method, called generation quality driven merging. Compared with previous methods (Wortsman et al., 2022; Matena and Raffel, 2022), the proposed method uses generation quality as the indicator to determine the merging weight and can relax the requirement of the labeled datasets.

Experimental results demonstrate that our proposed method achieves superior performance across multiple datasets compared to the best-performing individual models and exhibits higher generalization performance on out-of-distribution datasets. In particular, despite our proposed method not requiring a labeled dataset, our model merging approach can still outperform or approximate other strong baselines that leverage the label information.

2 Preliminary Experiments

In this section, our experiments focus on analyzing the merging of checkpoints from shared training trajectories during the pre-training phase. Our preliminary experiment mainly explores the following three aspects: (1) The influence of checkpoint proximity on the merging process; (2) The impact of the number of checkpoints on the merging outcome;

2.1 Experimental Setup

Datasets. We utilized C-Eval (Huang et al., 2023) and CMMLU (Li et al., 2024) as the experimental testbed to conduct preliminary investigations,

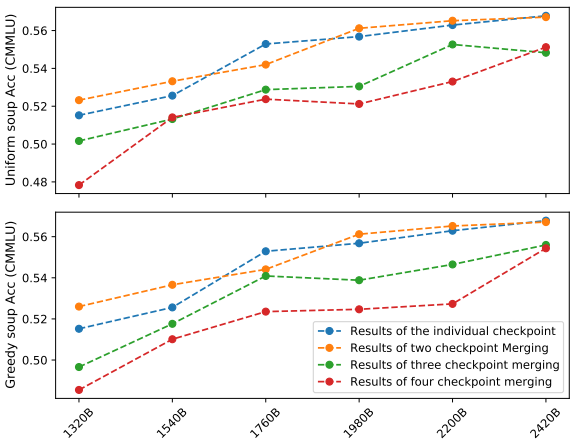

Figure 3: Performance derived from the merging of multiple checkpoints on CMMLU. Merging is conducted on intermediate checkpoints in Baichuan 2-7B using Uniform Soup and Greedy Soup.

where C-Eval consists of 13948 multi-choice questions spanning 52 diverse disciplines and four difficulty levels, and CMMLU is a comprehensive evaluation benchmark covering 67 topics that span from elementary to advanced professional levels.

Models. The checkpoints utilized in our pilot experiment are the 11 intermediate checkpoints of the 7B LLM released by Baichuan 2 (Yang et al., 2023), ranging from the 220 billion tokens checkpoint to the 2,640 billion tokens checkpoint.

2.2 How Checkpoint Proximity Affects the Model Merging?

To explore the influence of checkpoint proximity on model merging, we conduct a comprehensive assessment of the merged soup on both C-Eval and CMMLU. Specifically, for pairwise merging, 11 intermediate checkpoints from Baichuan 2-7B can yield a total of 55 (C_{11}^2) merged checkpoint combinations. We utilize in-context learning with 5 demonstrations for model reasoning in both C-Eval and CMMLU. We employ "Greedy Soup" (Wortsman et al., 2022) to conduct pairwise checkpoint merging and compare the performance difference between the merged soup and the best-performing individual checkpoint before merging.

Figure 2 presents the performance changes before and after checkpoint merging. The dark boxes indicate performance improvement. It is worth noting that the results indicate **there is a greater likelihood of performance enhancement when merging checkpoints that are adjacent in the pretraining phase**. For instance, merging ckpt-1320B with ckpt-1100B results in a 1.07% increase in accuracy compared to ckpt-1320B in CMMLU. Conversely, we can observe that, **as the distance during the pretraining phase increases, the performance of the merged checkpoint soup tends to decrease**. For instance, compared to ckpt-2420B, merging ckpt-2420B with ckpt-220B yields a substantial accuracy decrease of 32.04% in CMMLU. The same trend is also evident in the C-Eval dataset, presented in Appendix Figure 5.

2.3 How Checkpoint Numbers Affect the Merging?

Drawing from the findings of the aforementioned experiments, we impose a restriction on the merging of checkpoints, limiting it to checkpoints that are saved contiguously. Another aspect deserving attention is determining the optimal number of checkpoints to be merged to achieve superior performance. To this end, we investigate the influence of the number of checkpoints on the model merging by incrementally increasing the number of checkpoints. Specifically, we employ "Uniform Soup" and "Greedy Soup" for merging checkpoints on both C-Eval and CMMLU. We incrementally extend from pairwise checkpoint merging to the merging of four checkpoints, subsequently evaluating the performance of the merged soup.

The outcomes of merging multiple checkpoints are presented in Figure 3. For a clearer and more intuitive presentation of the outcomes of merging multiple models, we showcase the results of merging checkpoints in the late stages of pretraining. All the merged soups in the figure represent the merging of consecutive checkpoints before the specified point. It can be observed that **the pairwise merging of adjacent checkpoints generally leads to better outcomes compared to the individual checkpoint**.

Additionally, **the performance of merging three or four checkpoints is weaker than that of merging two checkpoints**. Meanwhile, merging three or four checkpoints typically leads to a performance drop, often even below that of the individual checkpoint before merging. For instance, the combination of four checkpoints using Greedy Soup and Uniform Soup resulted in a maximum performance drop of 3.60% and 3.69% respectively, compared to the best individual checkpoint on CMMLU. We observe the same trend in C-EVAL, as shown in the Appendix Figure 6.

3 Methodology

In this section, we first illustrate the formulation of checkpoint merging. Then, we introduce the implementation of our method, which can effectively and efficiently determine the merging weight.

3.1 Checkpoint Merging

When conducting LLM pretraining, we have already saved multiple checkpoints at the time t, denoted as $\{\theta_1, \theta_2, ..., \theta_t\}$. The linear combination of these multiple checkpoints in parameter space is referred to as "Checkpoint Soup" and can be represented as:

$$\widetilde{\theta}_t = \sum_{i=1}^{t} \lambda_i \theta_i \quad \textbf{s.t.} \sum_{i=1}^{t} \lambda_i = 1 \tag{1}$$

where $\lambda_i \in R$ denotes merging weight. Compared with ensemble on checkpoints (Dietterich, 2000), checkpoint merging is performed in the parameter space rather than the LLM output space, and meanwhile the checkpoint soup can be viewed as a new checkpoint along the training trajectory, which means the LLM can load the soup and continue pretraining.

According to the key findings in pilot experiments, we only focus on pairwise merging in this paper, therefore, Equation 1 can be reformulated as:

$$\widetilde{\theta}_t = \lambda_t \theta_t + (1 - \lambda_t)\theta_{t-1} \tag{2}$$

The key factor affecting the performance of the checkpoint after merging is the choice of merging weights. Besides, we provide a theoretical analysis that offers insights into why linear checkpoint merging can enhance model performance. We adopt three assumptions related to the actual characteristics of neural networks.

Assumptions 1 (Smoothness). The performance function of the LLM $f(\theta)$ is differentiable, and its gradient $\nabla f(\theta)$ is Lipschitz continuous:

$$\|\nabla f(\theta_1) - \nabla f(\theta_2)\| \leq K\|\theta_1 - \theta_2\|, \quad \forall \theta_1, \theta_2 \tag{3}$$

where constant $K > 0$.

Assumptions 2 (Non-Convexity and Quadratic Approximation). In LLMs, the performance function is typically non-convex. However, for two adjacent checkpoints, we can approximate their performance behavior using a quadratic function. Specifically, for $\delta = \theta - \theta_t$, we have

$$f(\theta) \approx f(\theta_t) + \nabla f(\theta_t)^\top \delta + \frac{1}{2}\delta^\top H_t \delta \tag{4}$$

where $||\delta||$ is small, H_t is the Hessian matrix.

Assumptions 3 (Bounded Hessian). The eigenvalues of the Hessian matrix at θ_t are bounded:

$$\lambda_{\min} I \preceq H_t \preceq \lambda_{\max} I \tag{5}$$

where $\lambda_{\min} \geq 0$ and $\lambda_{\max} > 0$ are constants, and I is the identity matrix.

Under the three assumptions mentioned above, we can derive the performance of the merged checkpoint satisfies $f(\widetilde{\theta}_t)$:

$$f(\widetilde{\theta}_t) \approx \lambda_t f(\theta_t) + (1 - \lambda_t) f(\theta_{t-1}) + \Delta \tag{6}$$

where Δ is defined as:

$$\Delta = (\lambda_t(1 - \lambda_t)K + \delta\frac{1}{2}\left[\lambda_t^2 + (1 - \lambda_t)^2\right]\lambda_{\max}) \times \|\theta_t - \theta_{t-1}\|^2 \tag{7}$$

The detailed proof is shown in Appendix B.

3.2 Generation Quality Driven Merging

Previous studies (Wortsman et al., 2022; Matena and Raffel, 2022) empirically demonstrate that it is better to assign a higher weight to the model exhibiting superior performance. In our work, we choose perplexity as the basis for weight allocation during checkpoint merging. Perplexity is a common metric used to assess the language model's generating capability by quantifying the uncertainty of a sequence, and can be denoted as:

$$\phi(x) = \exp\{-\frac{1}{t}\sum_{i=1}^{t} \log p(x_i|x_{<i})\} \tag{8}$$

where $p(x_i|x_{<i})$ represents the log-likelihood induced by the LLM and t denotes the sequence length. The reasons why we select perplexity are as follows: (a) Compared to previous methods that rely on labeled data to calculate accuracy (Wortsman et al., 2022) and approximate posterior information matrices (Matena and Raffel, 2022) as the basis for model merging weight allocation, the computation of perplexity does not require labeled data. (b) Several studies (Xia et al., 2023; Schaeffer et al., 2023) have demonstrated that linear or continuous metrics can produce smooth, continuous, and predictable changes in model performance, while nonlinear or discontinuous metrics may distort the performance of the model family, making it appear sharp and unpredictable. As a continuous measure, perplexity can effectively reflect the

quality of a model's generation of specific text. (c) During the pre-training process, certain tokens exhibit a trend of continuous learning (decreasing perplexity), while other tokens exhibit a trend of forgetting (increasing perplexity) or a stagnated trend. Additionally, LLMs with more computational power and capacity, tend to overfit to the subset tokens initially and subsequently generalize better (Xia et al., 2023). This implies that during the pre-training phase, LLMs may exhibit varying degrees of proficiency in learning different types of knowledge, and perplexity serves as an effective means of observing this phenomenon. Last but not least, there is a strong correlation between perplexity and performance across different checkpoints in the pre-training phase. The relationship between perplexity and performance at checkpoints is shown in the appendix A.3.

In detail, given a held-out unlabeled dataset $D = \{x_k\}_{k=1}^{n}$, and LLM checkpoints θ_t and θ_{t-1}, we can pass the dataset D through LLM with θ_t and θ_{t-1}, and obtain the perplexity $\phi(D|\theta_t)$ and $\phi(D|\theta_{t-1})$. Since a smaller perplexity indicates a better generation quality on the dataset D, the merging weight λ_t in the generation quality driven checkpoint merging can be denoted as:

$$\lambda_t = \frac{\frac{1}{\phi(D|\theta_t)}}{\frac{1}{\phi(D|\theta_t)} + \frac{1}{\phi(D|\theta_{t-1})}} \tag{9}$$

Correspondingly, The merged soup in the Equation 2 can be written as:

$$\widetilde{\theta}_t = \frac{\phi(D|\theta_{t-1})\theta_t + \phi(D|\theta_t)\theta_{t-1}}{\phi(D|\theta_t) + \phi(D|\theta_{t-1})} \tag{10}$$

Note that, the proposed generation quality driven merging is not confined to pairwise checkpoint merging, but can readily extend to merging multiple checkpoints.

4 Experiments

Our anticipation is that the merged soup will offer two primary benefits to the community. First, by merging several individual checkpoints in the pretraining trajectory, we expect the merged soup can achieve better performance on the target dataset, which we call "**In-distribution (IND)**" setting, since determining weight and testing the merged soup use the data from the same distribution. Second, the merged soup is also expected to showcase strong performance in the "**Out-of-distribution (OOD)**" setting, in other words, determining weight and testing the merged soup are respectively applied to datasets originating from different distribution. We conduct evaluations on multiple benchmarks to assess the performance of the merged soup in both in-distribution and out-of-distribution scenarios. Additionally, we explore the effectiveness of our model merging method on models of different scales. Finally, we analyze the factors that influence checkpoint merging, i.e., data quantity and input paradigms.

4.1 Experiment Setup

Datasets. Besides CMMLU (Li et al., 2024), and C-EVAL (Huang et al., 2023), we further select five benchmark datasets as the testbed: GSM8k (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), MedMCQA (Pal et al., 2022), PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019).

Checkpoints Apart from the 11 checkpoints provided by Baichuan 2-7B (Yang et al., 2023), we incorporate 10 checkpoints of size 7B released by Deepseek (DeepSeek-AI et al., 2024) for experiments. The latter encompasses checkpoints ranging from 200 billion tokens to 2000 billion tokens. We also utilize Pythia (Biderman et al., 2023) checkpoints of varying scales, ranging from 70M to 2.8B. Note that, based on the previous findings, all subsequent experiments are constrained to pairwise merging of adjacent checkpoints.

Baseline Merging Methods: In experiment, we compare our proposed method with the following strong baselines: (1) **Individual Checkpoint** To better showcase the performance changes after model merging, we report the performance of individual checkpoints before merging. Specifically, we define the average performance of all individual models before merging as **Avg.ckpt**, and the best performance achieved by the individual checkpoint before merging as **Best.ckpt**. (2) **Uniform Soup** (Wortsman et al., 2022) is a straightforward approach that takes the average of weights from all checkpoints. (3) **Greedy Soup** (Wortsman et al., 2022) sequentially adds models to the model soup and retains them in the soup if the accuracy on the held-out data does not decrease. (4) **Fisher-Weighted Averaging** (Fisher) (Matena and Raffel, 2022) is a method based on the Laplace approximation, where each checkpoint's posterior is approximated as a Gaussian distribution whose precision matrix corresponds to its Fisher information. (5) **Regression Mean** (RegMean) (Jin et al., 2023) is a method guided by weights that mini-

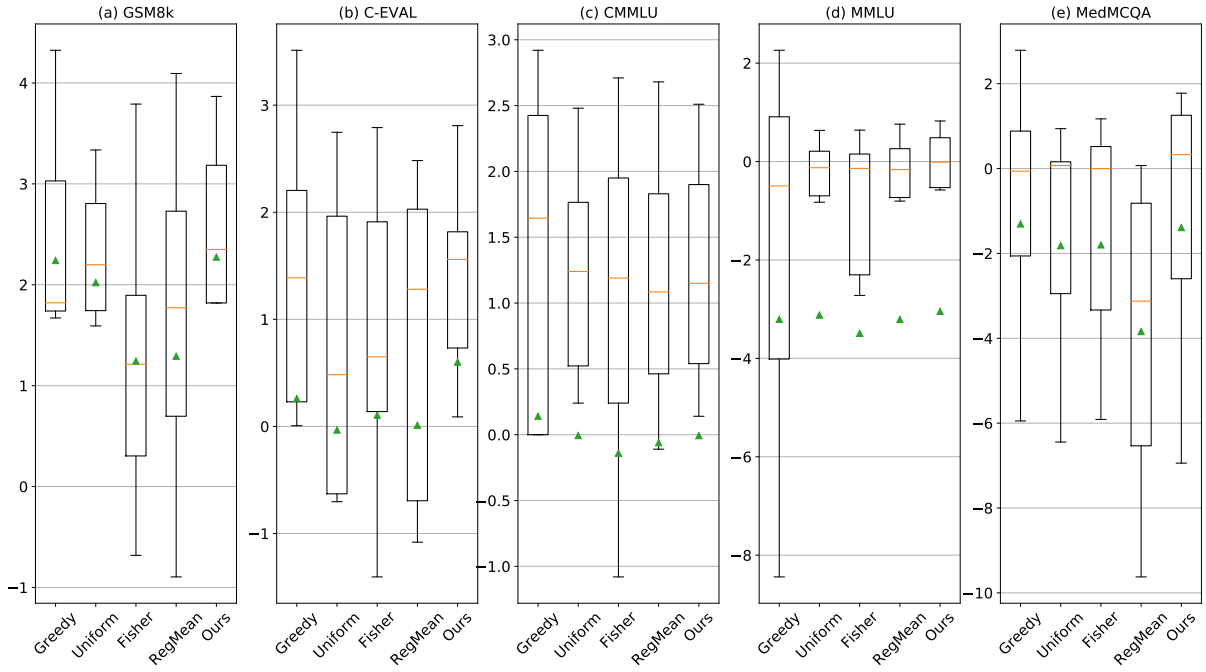

Figure 4: Relative performance drop (%) of soups obtained by pairwise checkpoint merging compared to the **Best.ckpt**. Positive values indicate performance improvement after merging. Box plots summarize the merged performance of 11 checkpoints from Baichuan 2-7B on GSM8k, C-Eval, CMMLU, MMLU and MedMCQA (from left to right). Green triangles indicate mean values and the orange lines represent the median values.

mize prediction differences between the merged soup and the individual models. It is worth noting that Uniform Soup is a data-free method, and our proposed method, as well as RegMean, only requires an unlabeled dataset. However, the Fisher and Greedy Soup depend on labeled datasets to compute approximate posterior information matrices and accuracy scores in order to assign weights to the different checkpoints.

4.2 Checkpoint Merging in the IND setting

The primary goal of checkpoint merging is to enhance the performance of the merged soup, without the need for continuous pretraining. We initially test the performance of pairwise merging of 11 checkpoints for each of the five tasks. Only focusing on merging adjacent checkpoints, we have 10 combinations of checkpoints in total. Figure 4 illustrates the relative performance drop of various merging methods on Baichuan 2-7B with respect to the best performance achieved by the individual checkpoint before merging (Best.ckpt). From the figure, we observe significant differences between merging methods, with our proposed method demonstrating superior and more stable performance. For instance, compared to the Best.ckpt, the ten merged checkpoints obtained by using our method achieved an average improvement of 2.22% on GSM8k. Furthermore, on the C-EVAL dataset, our method demonstrates superior performance over Uniform Soup, Greedy Soup, Fisher, and RegMean methods, with improvements of 0.34%, 0.63%, 0.49%, and 0.59%, respectively. Meanwhile, our checkpoint merging method shows positive values on most of the merged soups. Furthermore, despite Greedy Soup and Fisher requiring labeled datasets for merging, our method still achieves comparable or superior performance and demonstrates stronger stability. We also note that due to potentially significant performance discrepancies between adjacent checkpoints during merging, especially in the early stages of LLM pretraining, there might be a resultant performance drop in the merged soup.

Table 1 presents the results of merging Deepseek 7B checkpoints from different pre-training stages on C-Eval using various merging methods. It is noted that in the early stage of pre-training (ckpt-200B to ckpt-600B), the merged soups face challenges in achieving better results, primarily attributed to the sharp decline in loss during the early stage of model pre-training, resulting in significant differences in the distribution of LLM parameters. In the later stages of pre-training, merging meth-

Merged Checkpoint	Avg.ckpt	Best.ckpt	Uniform Soup	Greedy Soup	Fisher	RegMean	Ours
ckpt-200B & ckpt-400B	27.26	**29.31**	26.19	25.95	28.29	25.98	27.62
ckpt-400B & ckpt-600B	28.83	**29.31**	27.40	27.56	26.90	28.63	27.52
ckpt-600B & ckpt-800B	29.23	30.12	29.54	30.28	29.03	**30.72**	29.71
ckpt-800B & ckpt-1000B	31.15	32.17	32.52	32.57	32.29	32.81	**33.84**
ckpt-1000B & ckpt-1200B	33.02	33.87	37.16	37.79	**39.18**	37.64	38.41
ckpt-1200B & ckpt-1400B	36.33	38.80	**41.69**	40.84	40.37	40.43	41.37
ckpt-1400B & ckpt-1600B	39.10	39.40	41.26	40.70	40.24	39.55	**41.46**
ckpt-1600B & ckpt-1800B	41.23	43.05	41.41	41.45	42.78	41.98	**43.34**
ckpt-1800B & ckpt-2000B	43.70	44.36	44.61	44.70	44.81	43.95	**45.36**
Average Result	34.43	35.60	35.75	35.76	35.99	35.75	**36.51**

Table 1: In-distribution performance when merging 10 checkpoints of Deepseek 7B on GSM8k. Uniform Soup, Greedy Soup, Fisher, and Regmean are the model merging methods used for comparison.

Datasets	Greedy	Fisher	Regmean	Ours
CMMLU	56.3/56.6	56.5/56.2	56.8/56.6	56.7/56.7
MMLU	54.8/55.0	54.2/53.1	54.2/54.6	54.7/54.9
GSM8k	24.0/23.7	23.9/24.0	23.7/24.3	24.3/24.0
$\Delta(\downarrow)$	0.8	1.5	1.2	**0.5**

Table 2: Out-of-distribution performance when merging Baichuan 2-7B ckpt-2200B and ckpt-2420B on the C-Eval datasets. The data on the left and right sides represent the performance of checkpoints merged on the IND dataset and the C-EVAL dataset, respectively. Δ denotes the total difference in performance between IND and OOD on these three out-of-domain datasets.

ods tend to achieve more noticeable improvements relative to the best-performing individual checkpoint before merging. The results show that our merging algorithm attains the optimal average performance among the 10 model pairs, with improvements of 2.08%, 0.91%, 0.76%, 0.75%, 0.52%, and 0.76% compared to the Avg.ckpt, Best.ckpt, Uniform Soup, Greedy Soup, Fisher, and RegMean, respectively.

4.3 Checkpoint Merging in the OOD setting

Having established the relatively superior performance of our algorithm in in-distribution scenarios, we now turn our attention to another question, namely, Can the effectiveness of the merged soup, as determined within a specific dataset, generalize well when applied to a distinctly different dataset? Specifically, We merge the checkpoint using the Chinese dataset C-EVAL. We evaluate its performance on CMMLU, MMLU, and GSM8k, comparing it to the performance of checkpoints merged separately on these three datasets. We select ckpt-2200B and ckpt-2420B from Baichuan 2-7B for merging, since merging checkpoints from the later stages of pre-training typically leads to stable performance improvements.

The results are displayed in Table 2. From the table, we can find that despite the model merging being dependent on C-EVAL, the merged checkpoint still exhibits strong performance on other datasets. This suggests that the merging of checkpoints does not compromise their generalizability. Besides, compared to other merging methods, our approach shows the smallest average absolute performance difference between IND and OOD. This indicates that our merging method is least affected by the merging dataset and is more likely to be optimal across different domains. The checkpoint obtained from our merging method demonstrates stronger generalization capabilities.

4.4 Sensitivity Analysis

Checkpoint Merging on Models of Different Scales. In this part, we explore the impact of LLM parameter size on the effectiveness of our proposed model merging method. We perform checkpoint merging on Pythia models of different sizes and evaluate the performance of the merged models on the PIQA and WinoGrande datasets. Experiments show that the performance improvement of our proposed model merger is more stable across parameter sizes ranging from 70M to 2.8B. The results and analysis are presented in Appendix A.4.

The Impact of Checkpoint Merging on Model Supervised Fine-Tuning. After confirming that our merging method can obtain a superior and more stable checkpoint in terms of performance, we explored whether the merged checkpoint can be generalized to post-training scenarios. Specifically, we conduct an SFT experiment on the Alpaca dataset, where we merged Deepseek-1800B and Deepseek-

2000B based on the GSM8k dataset. The results are presented in Table 3, which demonstrates that the merged checkpoint serves as a better starting point.

The Impact of Data Quantity on Checkpoint Merging. In actual situations, the number of available data also is a noteworthy concern, aside from cases where obtaining data labels is not possible. Thus, we conduct a detailed examination of the influence of data quantity on model merging. Utilizing GSM8k for checkpoint merging, we examine how the size of the sample influences the performance of the merged soup. Our investigation covered performance variations in both IND and OOD. We designate C-Eval, CMMLU, MMLU, and MedMCQA as OOD datasets, showcasing the average performance of the merged soup on these datasets. Table 7 shows that our merging method remains effective in both IND and OOD scenarios even when employing only 1/4 of the data for merging. Moreover, across different data quantities, the maximum performance change on the IND is 0.45%, and the average performance change on the OOD datasets is only 0.03%. This indicates that our method can maintain consistent performance in situations with limited available datasets, and it is not sensitive to the quantity of available data.

The Impact of Calculating Perplexity with Different Input Paradigms on Checkpoint Merging. The superior performance exhibited by LLMs on many downstream tasks relies on their in-context learning capability. Since the calculation of perplexity can be influenced by context, a noteworthy question is how should we compute perplexity. Different forms of context can be categorized into three types: (a) Original input, which includes held-out unlabeled instances x_k, is defined as "Raw-input". (b) Zero-shot, which includes task instructions and x_k, is defined as "Zero-Shot-Input". (c) Few-shot, which includes few demonstrations, task instructions, and x_k, is defined as "Few-Shot-Input". We merge Baichuan 2-7B ckpt-2200B and ckpt-2420B checkpoints across a series of datasets. The results in Appendix Table 8 indicate that compared to calculating perplexity based solely on raw input, using Zero-Shot-Input and Few-Shot-Input to calculate perplexity can enhance the performance of the merged model. For instance, in the case of Few-Shot-Input, the average performance across five datasets increased from 44.12% to 44.73%. Additionally, the performance improvement of Few-Shot-Input containing demonstrations is relatively small, which may be attributed to the reverse impact of the information within the demonstrations on the perplexity.

Datasets	Deepseek-2000B	Ours
PIQA	80.30	**80.90(+0.60)**
Hellswag	71.15	**71.46(+0.31)**
Winogrande	80.30	**80.90(+0.60)**

Table 3: Performance comparison of Deepseek-2000B and the checkpoints merged from Deepseek-1800B and Deepseek-2000B based on our method after supervised fine-tuning on the Alpaca dataset.

Fine-grained Analysis of the Instance Level Performance Before and After Checkpoint Merging. An issue worth exploring is how the performance enhancement observed on specific datasets manifests at the dataset level after model merging. To explore this question, we conduct a fine-grained analysis of the performance of the models before and after merging on the instance level. Specifically, we select ckpt-2200B and ckpt-2420B from Baichuan 2-7B for merging and analyzing the similarities and differences in the performance of the model on five datasets before and after the merger. Experiment results indicate that the performance improvement brought about by model merging may stem from its inheritance of the performance of the models before merging. The results and analysis are presented in the Appendix A.5.

5 Conclusion

This paper explores the reduction of LLM pretraining consumption without raising computational costs through the adoption of a checkpoint merging approach. We first explore the characteristics of checkpoint merging through some pilot experiments. Subsequently, we propose a simple and almost free lunch approach that determines the merging weights based on the generation quality. Through extensive experiments, we demonstrate that our method outperforms the best-performing individual model on multiple datasets and exhibits superior performance and enhanced stability compared to other merge methods. Furthermore, our method also demonstrates higher generalization performance on out-of-distribution datasets. Thus, using generation quality as an indicator for LLM checkpoint merging is a promising avenue for exploration.

Limitations

We note that perplexity may not be a reliable metric for evaluating the quality of text, as it is sensitive to the length of the text. Specifically, the perplexity of short text is likely to be much higher than that of long text. Several prior works (Zhang et al., 2021; Meister et al., 2023) have also shown that neither low nor high perplexity are direct indicators of text quality. Therefore, a more reliable indicator of text quality would be highly beneficial. Additionally, we observe that when the continual pre-training of LLM shows a performance decline on a specific dataset, leveraging model merging methods to merge adjacent checkpoints during this trend often struggles to yield performance improvements.

References

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. 2021. Swad: Domain generalization by seeking flat minima. *Preprint*, arXiv:2102.08604.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *arXiv preprint arXiv:1710.03282*.

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. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, :, 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, A. 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. *Preprint*, arXiv:2401.02954.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg. Springer-Verlag.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. 2024. Llmcarbon: Modeling the end-to-end carbon footprint of large language models. *Preprint*, arXiv:2309.14393.

Harvey Fu, Qinyuan Ye, Albert Xu, Xiang Ren, and Robin Jia. 2023. Estimating large language model capabilities without labeled test data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9530–9546, Singapore. Association for Computational Linguistics.

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

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient BERT pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3774–3784, Dublin, Ireland. Association for Computational Linguistics.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. *Preprint*, arXiv:2307.13269.

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*.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. Dataless knowledge fusion by merging weights of language models. *Preprint*, arXiv:2212.09849.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2023. Solar 10.7b: Scaling large language models with simple yet effective depth up-scaling. *Preprint*, arXiv:2312.15166.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023. Deep model fusion: A survey. *Preprint*, arXiv:2309.15698.

Ziming Liu, Shenggan Cheng, Haotian Zhou, and Yang You. 2023. Hanayo: Harnessing wave-like pipeline parallelism for enhanced large model training efficiency. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '23, New York, NY, USA. Association for Computing Machinery.

Michael Matena and Colin Raffel. 2022. Merging models with fisher-weighted averaging. *Preprint*, arXiv:2111.09832.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally typical sampling. *Preprint*, arXiv:2202.00666.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa : A large-scale multi-subject multi-choice dataset for medical domain question answering. *Preprint*, arXiv:2203.14371.

Boris Polyak and Anatoli B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *Siam Journal on Control and Optimization*, 30:838–855.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.

Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. 2023. Diverse weight averaging for out-of-distribution generalization. *Preprint*, arXiv:2205.09739.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Sunny Sanyal, Atula Neerkaje, Jean Kaddour, Abhishek Kumar, and Sujay Sanghavi. 2023. Early weight averaging meets high learning rates for llm pre-training. *Preprint*, arXiv:2306.03241.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *Preprint*, arXiv:2304.15004.

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.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

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

Joachim Utans. 1996. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press*, pages 133–138. Citeseer.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *Preprint*, arXiv:2401.10491.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. *Preprint*, arXiv:2205.12410.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Ves Stoyanov. 2023. Training trajectories of language models across scales. *Preprint*, arXiv:2212.09803.

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. *Preprint*, arXiv:2309.10305.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *Preprint*, arXiv:2311.03099.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.

A Appendix

A.1 Related Work

Model merging is an emerging trend in recent research. Unlike traditional model ensemble techniques, which combine the outputs of multiple models to enhance the overall performance of a system. Model merging aims to combine multiple models into a single model with diverse or superior capabilities. It has been demonstrated that model merging can enhance the performance, robustness, and generalization of models (Li et al., 2023). A series of methods for model merging has been proposed in recent years. In detail, Wortsman et al. (2022) propose "Model Soup" that averaging of weights across numerous models without incurring any additional inference or memory costs. Similarly, Cha et al. (2021); Ramé et al. (2023) delve into the utilization of weighted averaging for models generated from different configurations, aiming to improve the out-of-distribution generalization. Matena and Raffel (2022) propose an alternative merging process aimed at overcoming the limitation of simple weight averaging, taking into account potentially varying weights' importance. Jin et al. (2023) proposed a dataless knowledge fusion method that merges models in their parameter space, guided by weights intended to minimize prediction discrepancies between the merged model and the individual models. Furthermore, expecting the mere merging of entire model parameters, (Wang et al., 2022; Huang et al., 2024) employed the application of linear mathematical operations to adapter parameters, resulting in superior generalization performance. Although numerous effective model merging methods have been put forward, we notice a lack of attention paid to the utilization of model merging methodologies during the pretraining phase. In this paper, we merge checkpoints of LLM and propose a new method that leverages generation quality as the indicator to determine the merging weight.

A.2 Pilot Experiments performance on C-Eval Dataset

Checkpoint Proximity Affects Model Merging on C-Eval Dataset. We first demonstrate the impact of checkpoint proximity on model merging performance using the C-Eval (Huang et al., 2023) dataset. Figure 2 presents the performance changes before and after checkpoint merging on C-Eval dataset. It is worth noting that merging checkpoints that are adjacent in the pretraining phase is more likely to result in performance enhancement (e.g., merging ckpt-1540B with ckpt-1320B can notably improve 2.14% in accuracy compared to ckpt-1540B in C-Eval). However, as the distance between checkpoints during the pre-training phase increases, the performance of the merged checkpoints tends to decline (e.g., compared to ckpt-2420B, merging ckpt-2420B with ckpt-220B yields a substantial accuracy decrease of 30.05% in C-Eval), which is consistent with the observations made on the CMMLU dataset.

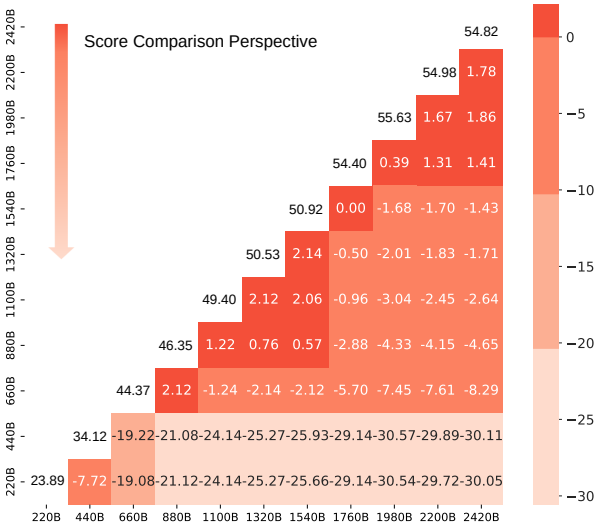

Figure 5: Performance Delta: 11 checkpoints Merged with Greedy Soup vs. Individual checkpoint Before Merging on C-Eval. The black numbers represent the original performance of the Checkpoint on two datasets.

Checkpoint Numbers Affects Model Merging on C-Eval. Figure 3 shows the results of merging multiple models on the C-Eval dataset. Similar to CMMLU, the pairwise merging of adjacent checkpoints generally leads to better outcomes compared to the individual checkpoint. For instance, in pairwise merging on Checkpoint, Greedy Soup and Uniform Soup achieved performance improvements of 2.65% and 2.36% on C-Eval, respectively. Conversely, merging three or four checkpoints tends to result in weaker performance compared to merging just two checkpoints. For instance, the combination of four checkpoints using Greedy Soup and Uniform Soup resulted in a maximum performance drop of 3.16% and 3.50% respectively, compared to the best individual checkpoint on C-Eval.

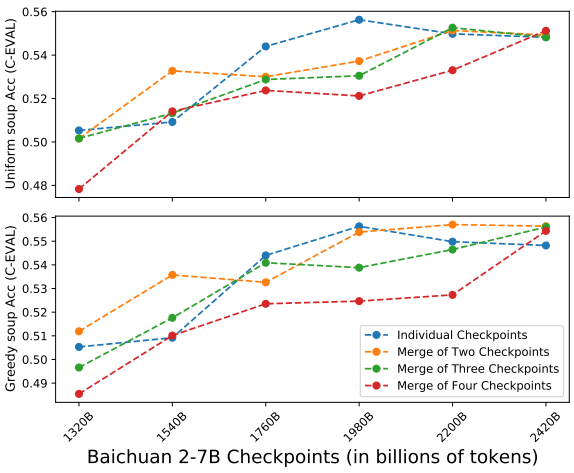

Figure 6: Performance derived from the merging of multiple checkpoints on C-Eval. Merging is conducted on intermediate checkpoints in Baichuan 2-7B using Uniform Soup and Greedy Soup.

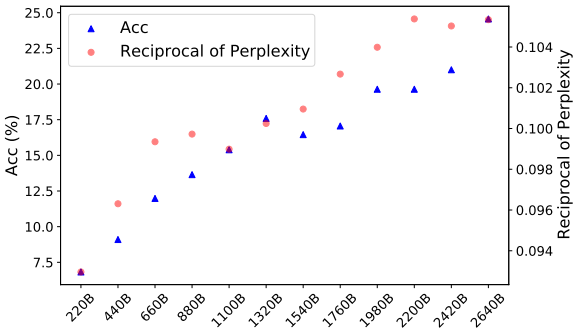

Figure 7: The relationship between the reciprocal of perplexity and accuracy on GSM8k.

Dataset	GSM8k	C-Eval	CMMLU	MMLU	Medmcqa
Coefficient	0.937	0.527	0.615	0.853	0.328

Table 4: The Spearman's rank correlation coefficients between the accuracy and the reciprocal of perplexity for 11 checkpoints on some benchmark datasets.

A.3 How the LLM's Performance on a Given Dataset Relates to its Proficiency in Generating the Same Dataset?

The adeptness of a LLM in text generation is indicative of its level of familiarity with textual information, an aspect intricately tied to its evaluation. Within this section, we examine the relationship between LLM's performance on a given dataset and its proficiency in generating the same dataset by calculating Spearman's rank correlation coefficient between accuracy and perplexity. Spearman's rank correlation coefficient, a nonparametric measure of rank correlation, evaluates the extent to which the association between two variables can be characterized by a monotonic function. A positive value indicates a positive correlation between the two variables, with a larger numerical value signifying a stronger correlation.

Perplexity is a common metric used to assess the reconstructive capability of LLM on text. Considering an auto-regressive LLM, we use $p(x_i|x_{<i})$ denote the log-likelihood induced by the LLM. Then we let $\phi(x) = \exp\{-\frac{1}{t}\sum_{i=1}^{M}\log p(x_i|x_{<i})\}$ denote the perplexity of sentence x, which quantifies the uncertainty of a sequence in relation to a specific LLM. Since a smaller perplexity indicates the language model is familiar with instances in the dataset and assigns high probability to these instances. Therefore, to visually illustrate the relationship between the generation quality and the performance of LLM, we present accuracy and the reciprocal of perplexity on GSM8k in Figure 7.

Meanwhile, we present the detailed Spearman's rank correlation coefficients between the accuracy and the reciprocal of perplexity for 11 checkpoints on several benchmark datasets in Table 4. From Figure 7 and Table 4, we find that: All Spearman's rank correlation coefficients are positive, which suggests **a positive correlation between the generation quality and the performance of LLM on the specified datasets**.

A.4 Checkpoint Merging on Models of Different Scales.

In this section, we conduct experiments with the Pythia (Biderman et al., 2023) model using different merging methods. We select the PIQA (Bisk et al., 2019) and Winogrande (Sakaguchi et al., 2019) datasets for evaluation, and consider four model sizes for Pythia: 70M, 410M, 2.8B. Table 5 and Table 6 present the performance of the merged models using different merging methods across the four model sizes on PIQA and WinoGrande, respectively. The results show that, across models of different sizes, our merging method consistently outperforms the label-independent methods, Uniform Soup and Regmean, and also achieves better average performance compared to the label-dependent methods, Greedy Soup and Fisher. Overall, the experiments show that, compared to other merging methods, our approach provides more stable performance improvements.

PIQA	70M	410M	2.8B	Average
Uniform	58.71	68.06	74.97	67.25
Greedy	58.71	68.06	74.76	67.18
Fisher	58.69	68.14	74.65	67.16
RegMean	58.96	68.32	74.72	67.33
Ours	59.42	68.17	74.81	**67.47**

Table 5: The results of merging Pythia models with different parameter sizes using various merging methods on the PIQA dataset.

Winogrande	70M	410M	2.8B	Average
Uniform	51.07	53.83	61.09	55.33
Greedy	51.07	53.83	60.85	55.25
Fisher	52.08	53.88	60.57	55.51
RegMean	51.97	53.76	60.89	55.54
Ours	51.98	54.06	61.25	**55.76**

Table 6: The results of merging Pythia models with different parameter sizes using various merging methods on the Winogrande dataset.

N	Perplexity	In-Distribution	Out-of-Distribution
1/4	9.49 / 9.51	24.18	49.32
2/4	9.56 / 9.57	23.96	49.34
3/4	9.51 / 9.53	23.81	49.35
Full	9.50 / 9.52	24.26	49.35

Table 7: Enumerating various sample sizes (N, the fraction of the dataset used for calculating generation quality) in merging ckpt-2200B and ckpt-2420B on the GSM8k dataset. We report the in-distribution performance and the average performance on out-of-distribution (OOD) datasets.

Dataset	Raw-Input	Zero-Shot-Input	Few-Shot-Input
GSM8k	**24.26**	24.18	**24.26**
C-Eval	54.72	**57.19**	55.55
CMMLU	56.73	**56.75**	56.66
MMLU	54.63	**54.74**	54.68
MedMCQA	30.25	**30.78**	30.54
Average Result	44.12	**44.73**	44.34

Table 8: Checkpoint merging based on perplexity calculated from different input paradigms. We merge Baichuan 2-7B ckpt-2200B and ckpt-2420B on the GSM8k, C-Eval, CMMLU, MMLU and MedMCQA datasets.

A.5 Fine-grained Analysis of the Instance Level Performance Before and After Checkpoint Merging.

In this section, we select ckpt-2200B and ckpt-2420B from Baichuan 2-7B for merging based on our methods and analyze the similarities and differences in the performance of the model on five datasets before and after the merging. The results are shown in Figure 8 to Figure 12. From the results, we can observe that there is a significant overlap between the correct and incorrect predictions of the checkpoint before and after merging. Additionally, when comparing the sizes of the independent regions corresponding to ckpt-2200B, ckpt-2420B, and the Merged-ckpt (represented by the orange, green, and purple parts on the Venn diagram respectively), we find that the merged checkpoint has the smallest independent region. Based on the above observations, we attribute the performance changes resulting from checkpoint merging to the merged checkpoint inheriting the performance of the pre-merged checkpoints. From Figure 11 and Figure 12, we observe that the independent region of the Merged-ckpt in the Venn diagram of positive samples in the GSM8k dataset and negative samples in the MedMcqa dataset is slightly higher than that of the checkpoint before merging (e.g., the independent region size of the Merged-ckpt on GSM8k is 62, while ckpt-2200B is 44 and ckpt-2420B is 49). These anomalous phenomena may be attributed to the dramatic performance improvements and declines before and after the merging.

B Proof for Linear Checkpoint Merging Bounds.

Under the assumptions in section §3.1, We derive the bounds of $f(\widetilde{\theta}_t)$ through the following steps:

1. Quadratic Approximation:

Expand $f(\widetilde{\theta}_t)$ around θ_t using the quadratic approximation Eq.4 with $\delta = \widetilde{\theta} - \theta_t = (1 - \lambda_t)(\theta_{t-1} - \theta_t)$:

$$\begin{aligned} f(\widetilde{\theta}_t) &\approx f(\theta_t) + \nabla f(\theta_t)^\top \delta + \frac{1}{2}\delta^\top H_t \delta \\ &= f(\theta_t) + (1-\lambda_t)\nabla f(\theta_t)^\top(\theta_{t-1} - \theta_t) + \\ &\quad \frac{1}{2}(1-\lambda_t)^2(\theta_{t-1} - \theta_t)^\top H_t(\theta_{t-1} - \theta_t) \end{aligned} \tag{11}$$

Similarly, expand $f(\widetilde{\theta}_t)$ around θ_{t-1} with $\delta = \widetilde{\theta} - \theta_{t-1} = \lambda_t(\theta_t - \theta_{t-1})$:

$$\begin{aligned} f(\widetilde{\theta}_t) &\approx f(\theta_{t-1}) + \nabla f(\theta_{t-1})^\top \delta + \frac{1}{2}\delta^\top H_{t-1} \delta \\ &= f(\theta_{t-1}) + \lambda_t \nabla f(\theta_{t-1})^\top(\theta_t - \theta_{t-1}) \\ &\quad + \frac{1}{2}\lambda_t^2(\theta_t - \theta_{t-1})^\top H_{t-1}(\theta_t - \theta_{t-1}). \end{aligned} \tag{12}$$

Construct an averaged approximation by forming a

convex combination of equations Eq.11 and Eq.12, where λ and λ_t are the respective weights.

$$\begin{aligned} f(\widetilde{\theta}_t) \approx & \lambda_t f(\theta_t) + (1-\lambda_t) f(\theta_{t-1}) \\ & + \lambda_t(1-\lambda_t)\left[\nabla f(\theta_t) - \nabla f(\theta_{t-1})\right]^\top (\theta_{t-1} - \theta_t) \\ & + \frac{1}{2}\Big[\lambda_t^2 (\theta_t - \theta_{t-1})^\top H_{t-1}(\theta_t - \theta_{t-1}) \\ & + (1-\lambda_t)^2 (\theta_{t-1} - \theta_t)^\top H_t (\theta_{t-1} - \theta_t)\Big] \end{aligned} \tag{13}$$

2. Bounding the Gradient Difference:

Under the assumption 1 in section §3.1, the term $L = \left[\nabla f(\theta_t) - \nabla f(\theta_{t-1})\right]^\top (\theta_{t-1} - \theta_t)$ can be bounded as:

$$\begin{aligned} L &\le \|\nabla f(\theta_t) - \nabla f(\theta_{t-1})\| \cdot \|\theta_{t-1} - \theta_t\| \\ &= \|\nabla f(\theta_t) - \nabla f(\theta_{t-1})\| \cdot \|\theta_t - \theta_{t-1}\| \\ &\le K\|\theta_t - \theta_{t-1}\|^2. \end{aligned} \tag{14}$$

3. Bounding the Hessian:

Under the assumption 3 in section §3.1, we can obtain:

$$(\theta_t - \theta_{t-1})^\top H_{t-1}(\theta_t - \theta_{t-1}) \le \lambda_{\max}\|\theta_t - \theta_{t-1}\|^2 \tag{15}$$

$$(\theta_t - \theta_{t-1})^\top H_t(\theta_t - \theta_{t-1}) \le \lambda_{\max}\|\theta_t - \theta_{t-1}\|^2. \tag{16}$$

4. Final Bound:

By combining equations Eq.13-16, we can obtain:

$$\begin{aligned} f(\widetilde{\theta}_t) \ge & \lambda_t f(\theta_t) + (1-\lambda_t) f(\theta_{t-1}) \\ & - \lambda_t(1-\lambda_t) K\|\theta_t - \theta_{t-1}\|^2 \\ & - \frac{1}{2}\left[\lambda_t^2 + (1-\lambda_t)^2\right]\lambda_{\max}\|\theta_t - \theta_{t-1}\|^2. \end{aligned} \tag{17}$$

Similarly, the upper bounds for the performance function be formalized as:

$$\begin{aligned} f(\widetilde{\theta}_t) \le & \lambda_t f(\theta_t) + (1-\lambda_t) f(\theta_{t-1}) \\ & + \lambda_t(1-\lambda_t) K\|\theta_t - \theta_{t-1}\|^2 \\ & + \frac{1}{2}\left[\lambda_t^2 + (1-\lambda_t)^2\right]\lambda_{\max}\|\theta_t - \theta_{t-1}\|^2. \end{aligned} \tag{18}$$

By combining equations Eq.17-18, we can obtain that the performance of the merged checkpoint satisfies:

$$\begin{aligned} f(\widetilde{\theta}_t) \approx & \lambda_t f(\theta_t) + (1-\lambda_t) f(\theta_{t-1}) \\ & \pm \left(\lambda_t(1-\lambda_t)K + \frac{1}{2}\left[\lambda_t^2 + (1-\lambda_t)^2\right]\lambda_{\max}\right)\|\theta_t - \theta_{t-1}\|^2. \end{aligned} \tag{19}$$

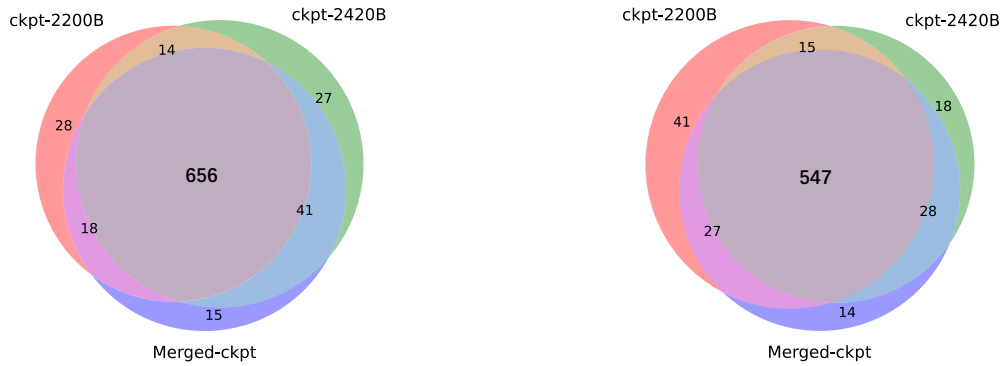

Figure 8: Degree of overlap between correct (left) and incorrect (right) samples on the C-Eval dataset before and after checkpoint merging.

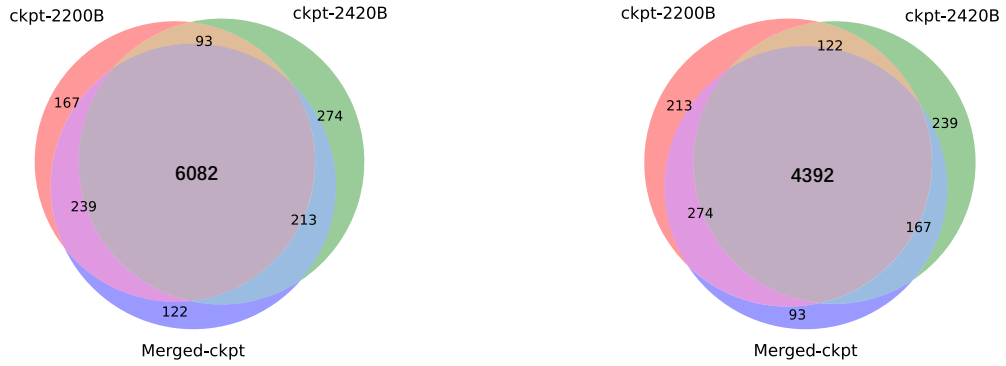

Figure 9: Degree of overlap between correct (left) and incorrect (right) samples on the CMMLU dataset before and after checkpoint merging.

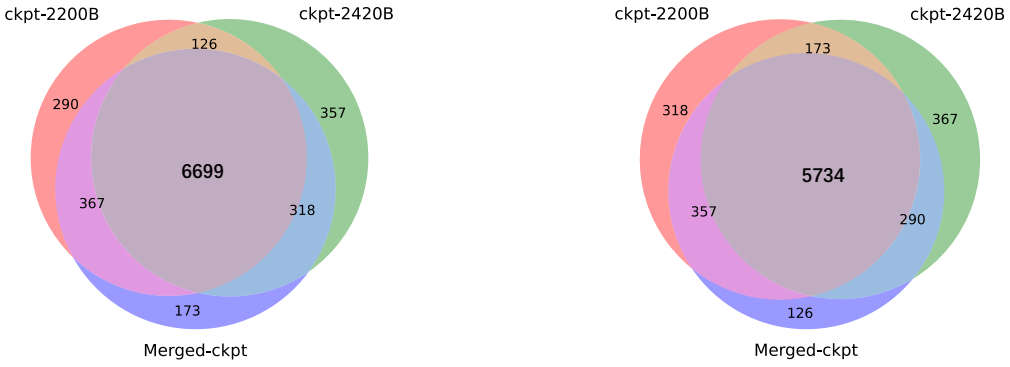

Figure 10: Degree of overlap between correct (left) and incorrect (right) samples on the MMLU dataset before and after checkpoint merging.

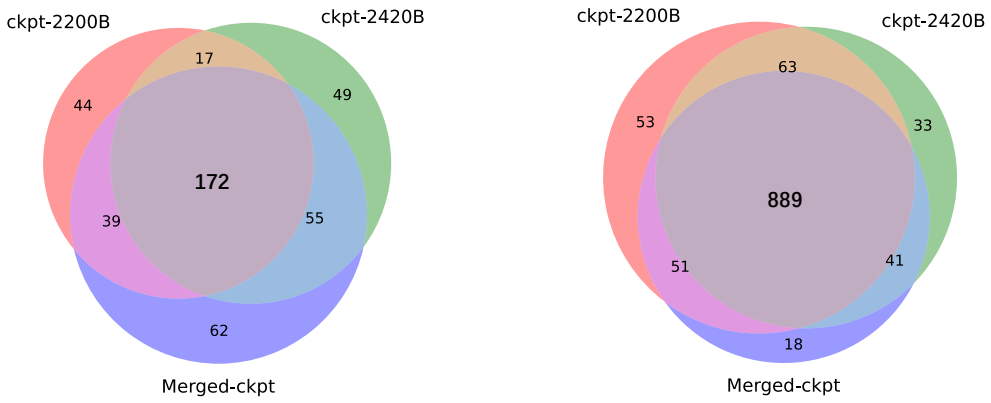

Figure 11: Degree of overlap between correct (left) and incorrect (right) samples on the GSM8k dataset before and after checkpoint merging.

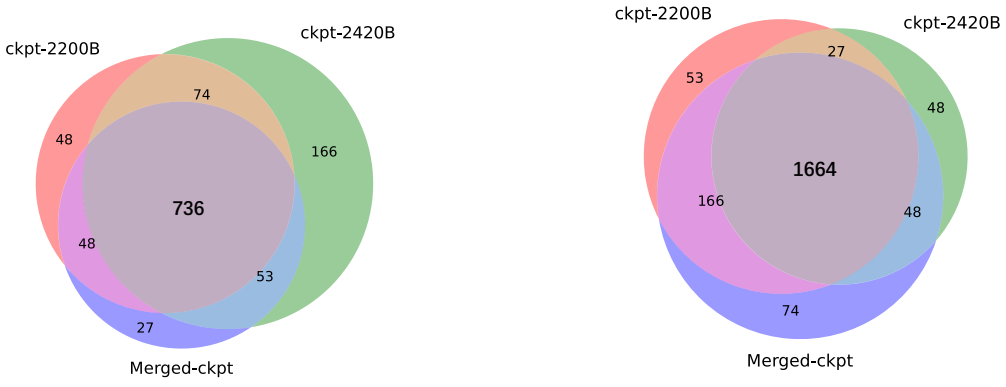

Figure 12: Degree of overlap between correct (left) and incorrect (right) samples on the MedMCQA dataset before and after checkpoint merging.