ExCP: Extreme LLM Checkpoint Compression via Weight-Momentum Joint Shrinking

Wenshuo Li [1] **Xinghao Chen** [1] **Han Shu** [1 2] **Yehui Tang** [1] **Yunhe Wang** [1]

Abstract

Large language models (LLM) have recently attracted significant attention in the field of artificial intelligence. However, the training process of these models poses significant challenges in terms of computational and storage capacities, thus compressing checkpoints has become an urgent problem. In this paper, we propose a novel Extreme Checkpoint Compression (ExCP) framework, which significantly reduces the required storage of training checkpoints while achieving nearly lossless performance. We first calculate the residuals of adjacent checkpoints to obtain the essential but sparse information for higher compression ratio. To further excavate the redundancy parameters in checkpoints, we then propose a weight-momentum joint shrinking method to utilize another important information during the model optimization, *i.e.*, momentum. In particular, we exploit the information of both model and optimizer to discard as many parameters as possible while preserving critical information to ensure optimal performance. Furthermore, we utilize non-uniform quantization to further compress the storage of checkpoints. We extensively evaluate our proposed ExCP framework on several models ranging from 410M to 7B parameters and demonstrate significant storage reduction while maintaining strong performance. For instance, we achieve approximately 70× compression for the Pythia-410M model, with the final performance being as accurate as the original model on various downstream tasks. Codes will be available at https://github.com/Gaffey/ExCP.

1. Introduction

Large Language Model (LLM) (Brown et al., 2020; Touvron et al., 2023; Wang et al.; Chowdhery et al., 2023; Team et al., 2023) has attracted the attention of the vast majority of academia and industry concentrated on Artificial Intelligence (AI). The current LLM can conduct daily conversations with humans, ask questions and answer questions, help humans extract information from articles and charts, and even complete professional-related tasks such as consultation and programming, which greatly improves the efficiency of human-computer interaction. Thousands of laboratories and companies are involved in the training of the LLMs. Computing power and storage have become key resources in the LLM era. Training an LLM requires up to thousands of GPUs or computing cards like TPUs or Ascends, and it is difficult to keep such a large computing cluster running completely smoothly. At the same time, researchers are also faced with the need to interrupt training at any time to adjust training data and hyperparameters. Sometimes it is even necessary to go back to earlier checkpoints to solve problems introduced during training. Therefore, frequent saving of checkpoints has become a must during the whole training process.

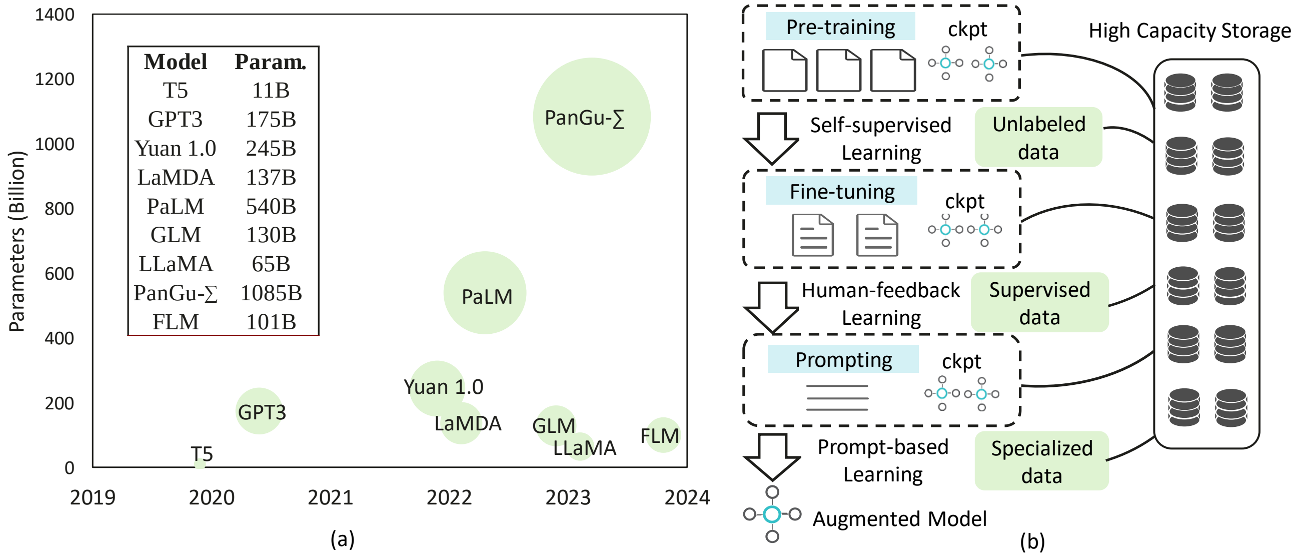
Take the open source model Pythia (Biderman et al., 2023) as an example, the checkpoint of the largest version Pythia-12B model takes more than 24GB to save. Not to mention the relevant momentum states of the optimizer. Adam optimizer requires twice the storage space of the weight. The training process of Pythia-12B saves 154 checkpoints which requires about 11TB storage, which would cost $5000 a month on a general cloud server to store these checkpoints. And this is just an entry-level scenario for large company. Conservative estimates suggest that the largest models of the most advanced LLMs, such as the GPT series and Gemini series, has the number of parameters on the order of hundreds to thousands of billions. Some publicly available data is shown Figure 1. Larger models also require more checkpoints and longer training time. So the total cost of storage for a cutting-edge LLM may grow to tens of millions of dollars.

In view of the above problems, compressing model checkpoints has become a very urgent need. Model compression

---

[1]Huawei Noah's Ark Lab [2]University of Science and Technology of China. Correspondence to: Xinghao Chen <xinghao.chen@huawei.com>, Yunhe Wang <yunhe.wang@huawei.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



Model	Param.
T5	11B
GPT3	175B
Yuan 1.0	245B
LaMDA	137B
PaLM	540B
GLM	130B
LLaMA	65B
PanGu-∑	1085B
FLM	101B

(a) (b)

Figure 1. The number of parameters of some LLMs and the general training process of LLMs. (a) Parameters of some recent LLMs, most of them contain billions of weights and keep getting larger in trend. (b) The training of LLMs consists of several stages with variety of schemes and data. A large quantity of checkpoints would be stored in each stage. Considering the magnitude of LLMs' parameters, extremely high capacity storage is needed for training of LLMs, which could cost tens of millions of dollars.

itself is not a new topic. The model size is compressed to reduce the storage occupied by checkpoints (Han et al., 2015; Hu et al., 2020; Eisenman et al., 2022; Chen et al., 2020; Jin et al., 2023; Agrawal et al., 2023) or compress the calculation amount of the model to improve the model's inference performance (Liu et al., 2017; Tang et al., 2020; Dettmers et al., 2022; Xiao et al., 2023; Chen et al., 2022; Shu et al., 2023; Wu et al., 2023). These researches have drawn attentions of researchers in the past ten years. However, previous checkpoints compression work concerns more about the size of weight checkpoints instead of the whole training states, so there is a lack of relevant researches on momentum states compression. In addition, the similarity of adjacent checkpoints should also be considered in the compression pipeline. This feature can improve the pruning ratio instead of simply reducing the final size using some encoding techniques.

In this paper, we propose a checkpoints compression framework that does not rely on training code and information. We calculate the residual value of adjacent checkpoints, apply weight-momentum joint pruning, and then non-uniformly quantize the weights and momentum states to extremely compress the checkpoints. Meanwhile, our residual compression strategy ensures that we can resume the training from compressed checkpoints nearly lossless. Our main contributions are as follows:

- We propose a checkpoints compression framework which contains residual calculation, weight-momentum joint pruning and non-uniform quantization. This framework makes full use of the characteristics of checkpoints compression, achieving almost lossless training recovery while achieving a high compression ratio.
- We derive a weight-momentum joint pruning method, and prove the convergence of the optimizer under this pruning method. This is the first work to our knowledge that jointly considers both weights and momentum states pruning.
- We conduct experiments on various models and evaluation benchmarks. Our compressed model achieves up to 70× nearly lossless compression on Pythia-410M model, which could largely reduce the storage of saving checkpoints.

2. Related Work

2.1. Large Language Model

Recently, the emergence of large language model and the corresponding strong capabilities in various natural language processing (NLP) applications have drawn widespread attention in research society. The demonstrated powerful abilities by the model scaling have furthermore increased the parameters of large language models. The remarkable work GPT3 (Brown et al., 2020) shows impressive performance on solving real-world NLP tasks. However, as shown in Table 1, the model contains 175 billion parameters and requires large amount of hardware resources to be trained and stored. A single training checkpoint of GPT3 can reach up to 2.3TB. Following large language models such as PaLM (Chowdhery et al., 2023) and LLaMA (Touvron et al., 2023) consume comparable or even more hardware resources. Due to the huge resource consumption and common training failures, the checkpoints of LLMs should

Table 1. The parameter and checkpoint size of part LLMs. High-capacity storage devices are essential for checkpoints for LLM training process.

Model	Param.	Storage
GPT3 (Brown et al., 2020)	175B	2.3TB
PaLM (Chowdhery et al., 2023)	540B	~7TB
LLaMA-70B (Touvron et al., 2023)	75B	1.0TB
PanGu-π (Wang et al.)	7B	99GB

be updated and stored frequently, which could occupy much more resources of the communication bandwidth and storage devices. Thus, the exploration of redundancy in LLM checkpoint is meaningful and necessary, which can save the memory consumption in great extent and make the training procedure more efficient and more affordable.

2.2. Compression Method

Data Compression Methods. Compression methods for efficient storage of data have been investigated for long decades. These previous methods can be categorized into two types, the lossy and the lossless. Lossy compression methods like JPEG (Rao & Hwang, 1996) and MP3 (Sterne, 2012) are widely used in the compression of image and video data which does not require precise restoration. Huffman coding (Huffman, 1952) is a classic lossless compression method, which statisticizes the frequency of the characters to get an optimized coding length according to different frequency of occurrence. The lossless compression method can be easily applied to the checkpoints of LLMs. However, the generalizability of the data compression method determines that the compression rate would be relatively low when applied to LLM checkpoints. Specialized compression method should be investigated and designed to achieve a higher compression rate for heavy intrinsic redundancy of LLM checkpoint.

Neural Network Compression Methods. The neural network compression methods have been explored by many work due to the increasing model size and computation resources. DeepCompression (Han et al., 2015) utilizes network pruning, quantization and huffman coding to obtain a compact neural network. Llm (Dettmers et al., 2022) and Smoothquant (Xiao et al., 2023) adopt the quantization to compress the large language models. These network compression methods could reduce the quantity or bit-width of parameters in neural networks but are often highly related to the training targets. Thus, these methods cannot be generally applied to compress the checkpoints with various task background. Moreover, re-training or finetuning is often necessary for compression methods like network pruning (Liu et al., 2017) and quantization-aware training (Esser et al., 2019), which could be extremely computationally expensive especially for large language models with huge training data and huge amount of network parameters.

Compression methods for checkpoints. As the deep neural network model getting larger and the training cost getting more expensive, some research work begin to focus on the compression of checkpoints. LC-Checkpoint (Chen et al., 2020) proposes a lossy compression scheme for checkpoint constructions on the assumption of SGD optimizer. Check-N-Run (Eisenman et al., 2022) applies differential and quantization for recommendation models. Delta-DNN (Hu et al., 2020) focuses on the storage of floating point numbers and records the differential of two neighboring versions. QD-Compressor (Jin et al., 2023) further develops a layer-based quantization and achieves higher compression ratio. When these methods applied on large-scale models of LLM, undesirable accuracy degradation would occur due to the uniform and constant quantization strategy during training procedure. Recent DynaQuant (Agrawal et al., 2023) tackles this issue by precisely compressing model parameters based on different contributions to the final result quality with an efficient dynamic quantization configuration and a quantization-aware delta encoding scheme. However, most of previous work focus on the compression of model parameters while ignoring the momentum states of optimizer, which occupy more memory storage and exist more redundancy in LLM checkpoints.

3. Our Method

A checkpoint $\mathcal{P}_t$ of a neural network during the tth training iteration generally contains the model weights $\mathcal{W}_t$ and parameters $\mathcal{O}_t$ of the optimizer momentum.

$$\mathcal{P}_t = \{\mathcal{W}_t, \mathcal{O}_t\}. \tag{1}$$

Saving checkpoints for T times during training leads to a series of checkpoints $\mathcal{P}$.

$$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_t \cdots, \mathcal{P}_T\}. \tag{2}$$

For the widely used Adam optimizer, the parameters with most significant storage cost are the first-order and second-order moments v_t, m_t, *i.e.*,

$$\mathcal{O}_t = \{v_t, m_t\}. \tag{3}$$

Note that some variables such as learning rate and weight decay *etc*. are also stored in the optimizer checkpoint, but can be simply neglected when compared with moments.

In the traditional pruning-related work, researchers only concern about the weights of models, since the main purpose of pruning is reducing the overhead of calculation and storage during the inference stage. However, when we turn to the checkpoint compression during the training process, the pruning of momentum is also important to reduce the total size of training checkpoints. Take the most general optimizer Adam used in LLM training as an example, it

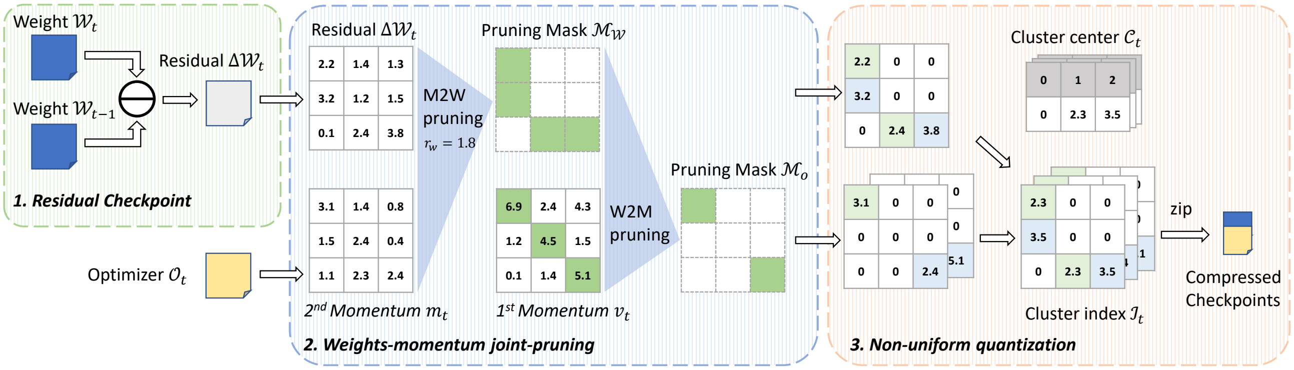


Figure 2. Framework of our proposed compression process. We calculate the residual $\Delta\mathcal{W}_t$ and apply joint-pruning on $\Delta\mathcal{W}_t$ and $\mathcal{O}_t$. Then we quantize them separately and save the cluster center $\mathcal{C}_t$ and cluster index $\mathcal{I}_t$.

saves the first-order and second-order moment of gradients which require double storage of weights. Therefore, we have to take both model weights and optimizer momentum states into consideration for extreme compression of model training checkpoints.

3.1. Residual Checkpoint

During the tth iteration of training, since we have already stored $t-1$ checkpoints[1] during previous training period, it is important to jointly utilize the temporal information of checkpoints to obtain more compact storage. The model weights will be updated upon previous ones according to the gradient during training, thus the difference between adjacent model weights is mostly to be sparse, which is more suitable for compression. In contrast, the momentum states stored in the optimizer checkpoints are the moving average of the first-order and second-order moments, which are only weakly related to the parameters in the previous checkpoint after updating for hundreds to thousands of steps, espectially for the first-order moment whose general $\beta_1 = 0.9$. So we do not apply residual calculation on optimizer momentum. The residual checkpoint $\Delta\mathcal{P}_t$ is defined as

$$\Delta\mathcal{P}_t = \{\Delta\mathcal{W}_t, \mathcal{O}_t\} = \{\mathcal{W}_t - \mathcal{W}_{t-1}, \mathcal{O}_t\}. \quad (4)$$

There is a comparison between the pruning on residual checkpoint and pruning on original checkpoint in Figure 3. We plot the histogram of the original weights, weights after direct pruning and weights after pruning on residual checkpoints. We find that pruning the residual checkpoint has almost no impact on the parameter distribution. This helps us to further prune the parameters.

3.2. Joint Weight-Momentum Pruning

Weight pruning is a common way to discard unimportant values while maintaining the performance to the maximum extent. For the checkpoint compression, we need to obtain the corresponding pruning masks for model weights and momentum states, which are denoted as $\mathcal{M}_w$ and $\mathcal{M}_o$, respectively. Intuitive way for pruning model weights and momentum states is to discard values with some pre-defined metric. However, this separate strategy leads to sub-optimal solution since there are strong relations between model weights and momentum states. Therefore, in this paper we propose a novel joint weight-momentum pruning method that obtains better performance for checkpoint compression.

Weight Pruning. For weights pruning, using the magnitude or the gradients of weights as an indicator is the common practice. There is a little difference between our weight pruning task and the general one. As we introduced in Section 3.1, we need to prune the residual values of weights of two adjacent checkpoints instead of the original value of weights. Thus we recommend to use the second-order moment of gradients of weights as an indicator, since they can represent the statistical average of the weight change during training process. We use the indicator to calculate the pruning threshold of each layer and the formula is shown below,

$$r_w = \frac{\alpha}{\sqrt{m_t}} \times \text{median}(\mathcal{W}), \mathcal{M}_w(i) = \mathbb{1}_{w_t(i) > r_w}. \quad (5)$$

In which $\mathcal{W}$ and m_t represents the weights and the second-order moment, respectively. α is a hyperparameter. After determining the pruning threshold of each layer, we prune the residual of weights to zero by magnitude for each layer.

Momentum Pruning. For momentum pruning, we use the first-order moment as an indicator to determine whether to prune this momentum states or not. We give a brief proof in the next section to explain why we choose it. Besides, if a specific location of weights is pruned, intuitively it is not important to preserve the corresponding momentum states. We prune the parameters of momentum following the formula below, in which β is a hyper-parameter.

$$r_o = \beta \times \text{mean}(v_t), \mathcal{M}_o(i) = \mathbb{1}_{v_t(i) > r_o \text{ and } \mathcal{M}_w(i)=1}. \quad (6)$$

---

[1] We give a detailed description about how to deal with previous checkpoints saved in residual format in Section 3.4

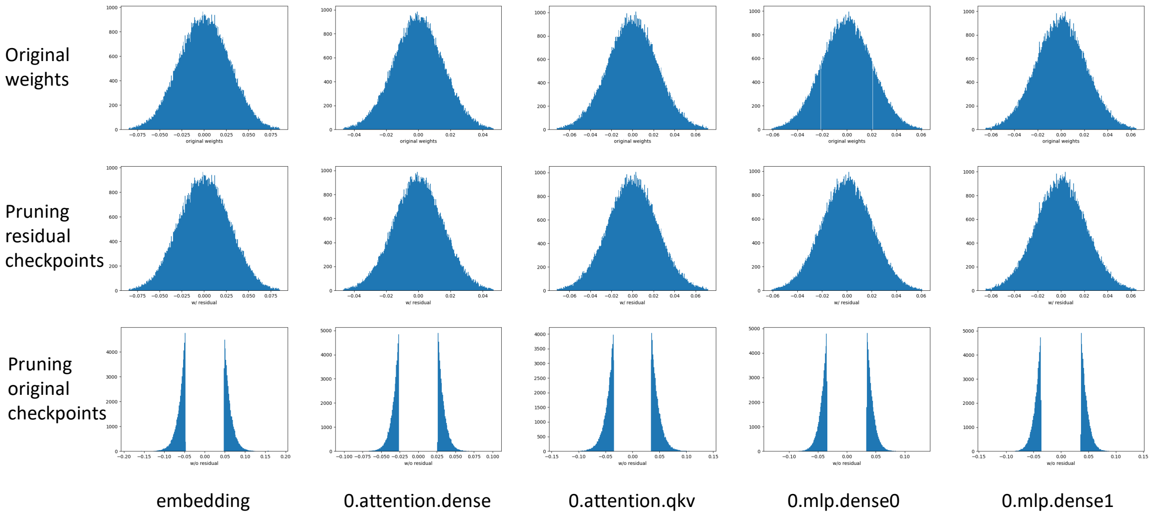


Figure 3. Weights distribution for original weights, pruning on residual checkpoints and pruning on original weights. We plot the histogram of random 100k non-zero weights of each case for clarity. The range of bins are bounded by (mean - 3 * std, mean + 3 * std) and 256 bins are used.

Convergence Analysis. Since we prune both the model weights and momentum states during training, it is important to analyze whether the whole training with checkpoint compression still have convergence guarantee.

Theorem 3.1. *According the convergence analysis in Adam (Kingma & Ba, 2014), assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. If we prune the moments with a mask $\mathcal{M}_o \in \{0, 1\}$ at iteration τ, Adam could also achieves the following guarantee, for all $T \geq 1$.*

$$\begin{aligned}\tilde{R}(T) \leq & \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\widehat{v}_{T,i}} \\ & + \frac{\alpha(1+\beta_1) G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \\ & + \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha} \sum_{i=1}^{d} \sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})} \sqrt{t} \\ & + \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (\sqrt{T\widehat{v}_{\tau,i}} - \sqrt{T\widehat{v}_{\tau,i}\mathcal{M}_o})\end{aligned} \tag{7}$$

Compared with the original convergence analysis of Adam (Kingma & Ba, 2014), the regret bound for our checkpoint compression method has an additional term:

$$\begin{aligned}\Delta\tilde{R}(T) &= \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (\sqrt{T\widehat{v}_{\tau,i}} - \sqrt{T\widehat{v}_{\tau,i}\mathcal{M}_o}) \\ &= \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (\sqrt{T\widehat{v}_{\tau,i}(1-\mathcal{M}_o)}).\end{aligned} \tag{8}$$

Since we only prune the values that v is relatively small, thus the regret bound is quite close to that of original training process.

Similar to the original optimization process of Adam, the average regret of our method also converges. Denote the regret bound of original Adam as $R(T)$, thus we have

$$\lim_{T\to\infty} \frac{\tilde{R}(T)}{T} \leq \lim_{T\to\infty} \frac{R(T) + \Delta\tilde{R}(T)}{T} = 0 \tag{9}$$

Therefore, our pruning method for momentum also achieves the following guarantee for all $T \geq 1$:

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right) \tag{10}$$

This indicates that our method also has good convergence rate as that of training without checkpoint compression. Detailed analysis can be found in Appendix A.

It should be noted that Sashank *et al*. (Sashank et al., 2018) point out the potential problem with the proof of convergence in (Kingma & Ba, 2014) and tremendous efforts (Shi & Li, 2021; Zhang et al., 2022; Défossez et al., 2020) have been taken for better analysis of convergence of Adam algorithm. In the above section, we provide an proof-of-concept analysis for the convergence of our checkpoint compression

Algorithm 1 Compressing process

```
Require: last reconstructed weight checkpoint $\hat{\mathcal{W}}_{t-1}$, original weight checkpoint $\mathcal{W}_t$, original optimizer checkpoint $\mathcal{O}_t$
    $\Delta\mathcal{W}_t \leftarrow \mathcal{W}_t - \hat{\mathcal{W}}_{t-1}$
    $\Delta\mathcal{W}_t^*, \mathcal{O}_t^* \leftarrow \mathrm{joint\_prune}(\Delta\mathcal{W}_t, \mathcal{O}_t)$
    $\mathcal{I}_t^{\mathcal{W}}, \mathcal{C}_t^{\mathcal{W}} \leftarrow \mathrm{quantize}(\Delta\mathcal{W}_t^*)$
    $\mathcal{I}_t^{\mathcal{O}}, \mathcal{C}_t^{\mathcal{O}} \leftarrow \mathrm{quantize}(\Delta\mathcal{O}_\mathrm{t}^*)$
    $\mathcal{P}'_t \leftarrow 7zip(\mathcal{I}_t^{\mathcal{W}}, \mathcal{C}_t^{\mathcal{W}}, \mathcal{I}_t^{\mathcal{O}}, \mathcal{C}_t^{\mathcal{O}})$
    save $\mathcal{P}'_\mathrm{t}$
```

framework. Incorporating newer convergence analysis of Adam into our framework should be feasible since we only modify the moments in a specific iteration. Extensive experiments also demonstrate the convergence of our proposed method.

3.3. Quantization

Besides pruning, quantization is also a common used method to compress the models or reduce the overhead of calculations. In our task, we only concern about the size of checkpoint, so we can choose non-uniform quantization method, which has better compression ratio but cannot accelerate the inference process.

Shown in Figure 2, we quantize weights and momentum states separately. We leave the pruned weights or momentum states to zero, and apply K-means algorithm on other weights or momentum states to cluster them to $2^n - 1$ cluster centers. Then we save the cluster centers $\mathcal{C}_t$ and cluster index $\mathcal{I}_t$.

3.4. Compressing and Reconstructing Checkpoints

Based on the methods we describe on the above sections, we here give a detailed introduction of our compressing and reconstructing process. During the whole training process, once we reach a saving node, we start our compression process independent of the main training process. We always keep a reconstructed version of the last checkpoint, for fast compression and training recovery. With this reconstructed version, the compressing process is described in Algorithm 1.

In which $\mathcal{I}$ and $\mathcal{C}$ represent the index and clustering center of non-uniform quantization, respectively. After we finish the compression of iteration t, we can either keep the original checkpoint $\mathcal{W}_t$ or reconstruct the weight checkpoint $\hat{\mathcal{W}}_t$ for the compression of next checkpoint or fast resumption from training crashed. We delete the other checkpoints $\hat{\mathcal{W}}_i, i \leq t-1$ to save the storage.

We can reconstruct a weight checkpoint $\hat{\mathcal{W}}_t$ from last saved/reconstructed weight checkpoint $\mathcal{W}_{t-1}/\hat{\mathcal{W}}_{t-1}$ and our saved compressed checkpoint $\mathcal{P}_t$ following the Algorithm 2.

Algorithm 2 Reconstructing process

```
Require: last reconstructed weight checkpoint $\hat{\mathcal{W}}_{t-1}$, compressed checkpoint $\mathcal{P}_t$
    $\mathcal{I}_t^{\mathcal{W}}, \mathcal{C}_t^{\mathcal{W}}, \mathcal{I}_t^{\mathcal{O}}, \mathcal{C}_t^{\mathcal{O}} \leftarrow \mathrm{unzip}(\mathcal{P}_\mathrm{t})$
    $\Delta\mathcal{W}_t^{Q*} \leftarrow \mathcal{C}_t^{\mathcal{W}}[\mathcal{I}_t^{\mathcal{W}}]$
    $\hat{\mathcal{O}}_t \leftarrow \mathcal{C}_t^{\mathcal{O}}[\mathcal{I}_t^{\mathcal{O}}]$
    $\hat{\mathcal{W}}_t \leftarrow \hat{\mathcal{W}}_{t-1}/\mathcal{W}_{t-1} + \Delta\mathcal{W}_t^{Q*}$
```

Algorithm 3 Reconstructing arbitrary checkpoints

```
Require: random seed $s$, compressed checkpoints $\mathcal{P}_i$, required iterations $t$
    $\hat{\mathcal{W}}_0 = \mathrm{init(s)}$
    while $i < t$ do
        $\hat{\mathcal{W}}_i \leftarrow \mathrm{recon}(\hat{\mathcal{W}}_{\mathrm{i-1}}, \mathcal{P}_\mathrm{i})$
    end while
```

Once we finished the whole training process, only the random seed for initialize weights and the compressed checkpoints are required to be saved, which are significantly smaller than the whole weights and optimizer checkpoint. If we want to reconstruct arbitrary checkpoints, we can follow the Algorithm 3.

4. Experiments

4.1. Models and Datasets

We conduct our experiments on ViT-L32 (Dosovitskiy et al., 2020), Pythia-410M (Biderman et al., 2023), PanGu-π-1B and PanGu-π-7B (Wang et al., 2023) models. The ViT-L32 model is trained and evaluated on ImageNet-1K dataset. As for LLMs, we train Pythia-410M on on a subset of the standard Pile (Gao et al., 2020) dataset. We used about 1/20 of the 300B tokens for this experiment. The PanGu-π series models are trained following the training details of their paper (Wang et al., 2023), which are trained on about 1.6 trillion tokens.

For evaluation, we use opencompass (Contributors, 2023) as the evaluation framework. We choose HellaSwag, ARC-easy, PIQA, C3, CSL and LAMBADA tasks to evaluate the performance of Pythia-410M, since these evaluation benchmarks are sensitive to model performance. And we evaluate the PanGu-π series models following their paper (Wang et al., 2023) which uses 11 benchmarks belonging to 4 categories, examination, knowledge, reasoning and understanding.

4.2. Implementation Details

During the training process, we save checkpoints every epoch for ViT-L32 and every 1000 iterations for LLMs, and compress the checkpoints in the meantime. We break our training process periodically, and then we resume the training from our compressed checkpoints until we finish

Table 2. Results of ViT-L32 model on ImageNet-1K dataset. CR represents the compression ratio. M2W pruning represents the Momentum-to-Weights pruning shown in equation 5. No check means that the weights are pruned with a setting threshold instead of $\frac{\alpha}{\sqrt{m_t}}$. And W2M pruning represents the Weights-to-Momentum pruning shown in equation 6. No check means that $\mathcal{M}_o(i) = \mathbb{1}_{v_t(i)>r_o}$.
* We estimate the results of the other work in terms of the checkpoint size of momentum being twice the weights.

Method	M2W pruning	W2M pruning	Top-1 Accuracy(%)	CR(Weights)	CR(Weights & Momentum)
baseline			71.36	1	1
CNR+			71.57	7.82	1.41*
QD+			71.24	16.31	1.45*
DynaQuant			71.82	26.19	1.47*
ExCP(Ours)			71.51	-	19.88
ExCP(Ours)	✓		71.53	-	25.54
ExCP(Ours)		✓	71.80	-	22.76
ExCP(Ours)	✓	✓	71.69	-	35.21

Table 3. Results of Pythia-410M models. We achieve almost lossless results while the storage is reduce by ~70×.

Model	Method	Size	Tasks						
			hellaswag	arc-e	piqa	C3	csl	lambada	Avg
Pythia-410M	Original model	4.53G	**32.52**	35.80	**62.13**	**37.21**	**53.75**	37.22	43.11
	Residual+7Zip	3.40G	**32.52**	35.80	**62.13**	**37.21**	**53.75**	37.22	43.11
	ExCP (Ours)	**0.06G**	31.95	**37.04**	62.62	36.22	52.50	**37.24**	42.93

the whole training process.

Unless otherwise specified, we set the α in equation 5 and β in equation 6 as $5e-5$ and 2.0 in our experiments, respectively. The weights except zero are non-uniformly quantized to $2^n - 1$ clustering center while the value zero occupies one center. And the bit number n is set as 4 in experiments. We combine two int4 number into one int8 number while saving. 7zip compression algorithm is used for further storage reduction.

4.3. Experimental Results

First we evaluate the effectiveness of our methods on ViT-L32 models to make a comparison with other checkpoint compression methods shown in Table 2. The accuracies of CNR+, QD+ and DynaQuant are all from Dynaquant (Agrawal et al., 2023). We follow the same setting as Dynaquant that we break and resume the training process from compressed checkpoints every 15 epochs. The results show that our method achieve better compression ratio which reachs more than 30×. Moreover, previous work do not compress the momentum of optimizer, which means that they achieve even less compression ratio while considering both weights and momentum. We also evaluate our joint pruning strategy. Using both M2W and W2M pruning achieve the best results.

Next we evaluate the performance of our methods on a relatively smaller LLM Pythia-410M. The results are shown in Table 3. The period of breaking is set to 5000 iterations. From the results, we can find that our methods achieve nearly lossless compression with ~70× compression ratio even the model itself is small enough in LLMs. Besides, our evaluation results on downstream tasks are even a little bit better than that of the original model. We think this is the improvement in generalization ability brought by the introduction of additional regularizers during the compression process.

The training loss curve and the comparison of storage of each checkpoint during the training process is shown in Figure 4. We plot the training loss every 100 iterations. We can find that our training curve fits well with the original one, with only a small loss increase. Meanwhile, we achieve $\sim 70\times$ compression in every node. When the learning rate is large, the stored checkpoint is relatively large. When the learning rate is small, the model size decreases since the model tends to be stable. This is related to our strategy of compressing residuals.

We also evaluate the PanGu-π-1B and PanGu-π-7B models in Table 4. Since we reproduce their results, the original models may differ slightly from the results in their paper. Our compressed models achieve $\sim 25\times$ storage reduce with negligible precision loss. Notice that the weights of PanGu-π series models are originally saved in float16 format, so the compression ratio would be less than that of Pythia series models.

We show a question & answer results for PanGu-π-1B model in Figure 5 to show the difference between our compressed model and the original model. In this example, our compressed model shows a better understanding to the limit

Table 4. Results of PanGu-π series models. We achieve almost lossless results while the storage is reduced by $\sim 25\times$.

Model		Size	Examination				Knowledge	Reasoning			Understanding			Avg
			C-Eval	CMMLU	MMLU	AGI-Eval	BoolQ	AX-b	PIQA	CSL	EPRSTMT	XSum	LCSTS	
PanGu-π-1B	Ori	14.64G	38.05	37.86	34.96	30.42	58.62	43.75	62.02	56.25	55.62	16.00	14.60	40.74
	Ours	0.59G	36.71	38.65	37.13	31.87	59.30	42.66	61.10	55.00	56.25	16.31	14.14	40.83
PanGu-π-7B	Ori	98.59G	59.91	59.97	61.84	54.04	64.59	56.88	77.31	63.12	90.00	19.59	16.61	56.71
	Ours	4.10G	61.32	60.14	62.37	55.11	68.44	52.90	77.91	63.75	90.00	19.24	16.77	57.09

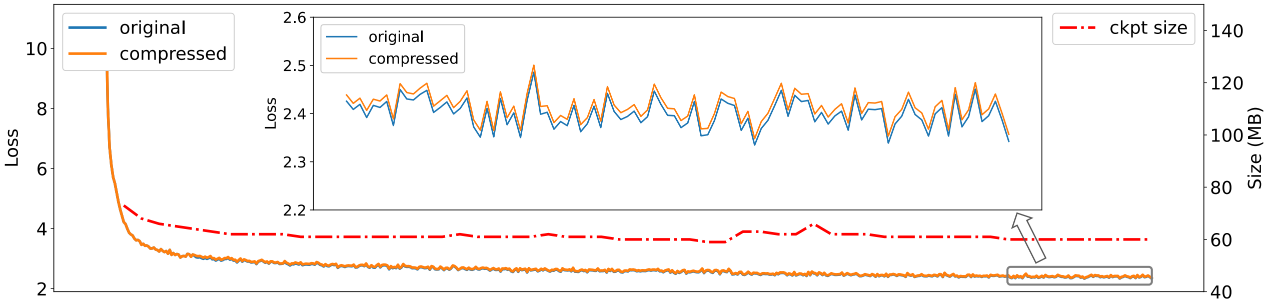


Figure 4. Comparison of training loss and checkpoint size between original models and our methods.

Table 5. Ablation study of our methods. Applying residual, joint-prune and quantization together achieves the best size while the average accuracy is almost lossless.

method			Size	Avg Acc
residual	prune	quant		
			4070M	43.11
✓			3484M	43.11
	✓		324M	29.95
		✓	492M	40.17
✓	✓		276M	42.92
✓		✓	493M	**42.94**
✓	✓	✓	**61M**	42.93

indoors and gives a better answer. It proves that our compressed model perfroms even better than the original one in some cases. More results are shown in the Appendix B.

4.4. Ablation Studies

We also do some ablation studies to show that every method in our compression pipeline is of vital importance. The results are shown in Table 5. Although calculating the residual of adjacent models cannot bring a significant storage reduce, it plays an important role in the whole pipeline. Directly pruning weights may harm the accuracy largely, while the residual of adjacent models does not have this problem. Joint-pruning and quantization on residual checkpoint separately compress the model by $15\times$ and $8\times$, respectively. Combining these two methods brings a large improvement to about $70\times$.

We explore the influence of different quantization bins. From Table 6, we can find that quantization below 4 bit cannot bring a significant storage reduce, so we choose 4 bit which achieves better performance-size trade-off. In some cases which extreme small checkpoint size is required, 2 bit could be used to further compress the checkpoints a little bit more.

Table 6. Ablation study of different quantization bins. We choose 4 bit in all other experiments since it achieves better performance-size trade-off.

Quant bins	Size	Avg Acc
2 bit	**43M**	42.46
4 bit	61M	**42.93**
8 bit	87M	42.90

Table 7. Comparison of different compression algorithms. We choose 7zip in all other experiments since it outperforms other algorithms.

	zip	rar	rar4	bz2	7z
size	73M	70M	69M	64M	**61M**

We also evaluate different compression algorithms to compact the final checkpoint files. The results are shown in Table 7. The 7zip compression with LZMA2 algorithm achieves the best compression ratio, which leads to about 20% less storage, and we apply 7zip with the ultra compression ratio on all other experiments.

5. Conclusion

In this paper, we discuss the extreme compression of LLM checkpoint. We propose a checkpoint compression framework which contains residual calculation, weights-momentum joint pruning and non-uniform quantization. We derive the criterion for weight-momentum joint-pruning and prove the convergence of the pruned momentum states.

Question: What kind of exercise do you suggest I do indoors?

Original PanGu-π-1B: If you are looking for a way to do some exercise in the comfort of your own home, there are several great exercises you can try. Some suggestions are: Squats: This is a great exercise for all levels, from beginner to advanced. You can do them by sitting on a chair with a squat rack at the back. Slowly lower your hips until your thighs are parallel to the ground, then push it back up. 2. Plank: Planks are a great core-strengthening exercise that can help improve your posture and lower your heart rate. Start by getting into a pushup, then a plank, and finally a layover. 3. Mountain climbers: Mountain climbers are a fun and challenging exercise that can help improve your strength and endurance. Start by bringing a heavy object, like a bag or bottle, into a vertical position. Climb the object up to the ceiling, then bring it back down to the ground. 4. Mountain climbers: Mountain climbers are a challenging and exciting exercise that can help improve your upper body strength. Start by bringing a heavy object, like a bag or bottle, into a vertical position. Climb the object up to the ceiling, then bring it back down to the ground. 5. Yoga: Yoga is a great way to improve your flexibility, strength, and balance. There are many online videos and apps that you can use to learn. Remember, it's important to start slowly and gradually increase the intensity of your workouts. If you feel uncomfortable with any of these exercises, you can always turn back to doing them at your own pace.

Compressed PanGu-π-1B: There are a lot of great exercises you can do indoors to help improve your fitness, such as: 1. Yoga: Yoga is a great way to improve flexibility, strength, and balance. There are many online videos and classes available that can help you learn the basics. 2. Bodyweight exercises: You can do bodyweight exercises such as push-ups, squats, lunges, and planks to build strength and improve your overall fitness. 3. Dancing: Dancing is a fun way to improve your cardiovascular fitness, coordination, and balance. There are many online tutorials and classes available that can help you learn the basics. 4. Yoga for beginners: If you're new to yoga, there are many beginner classes available that can help you learn the basics. 5. Pilates: Pilates is a great way to improve your core strength, flexibility, and balance. There are many online videos and classes available that can help you learn the basics. Remember to start slow and gradually increase the intensity and duration of your workouts as you get stronger and more confident.

Figure 5. Q&A example to show the difference between our compressed model and the original model.

Experimental results show the effectiveness of our methods. We compress Pythia-410M by $\sim 70\times$ while achieving nearly lossless results on down-stream evaluations.

In the future, we would try to extend the experiments to different tasks such as multi-modal large models and visual large models. And different types of neural networks including transformers, CNNs and RNNs would be taken into consideration.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agrawal, A., Reddy, S., Bhattamishra, S., Nookala, V. P. S., Vashishth, V., Rong, K., and Tumanov, A. Dynaquant: Compressing deep learning training checkpoints via dynamic quantization. *arXiv preprint arXiv:2306.11800*, 2023.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, X., Zhang, Y., and Wang, Y. Mtp: multi-task pruning for efficient semantic segmentation networks. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2022.

Chen, Y., Liu, Z., Ren, B., and Jin, X. On efficient constructions of checkpoints. *arXiv preprint arXiv:2009.13003*, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Contributors, O. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`, 2023.

Défossez, A., Bottou, L., Bach, F., and Usunier, N. A simple convergence proof of adam and adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eisenman, A., Matam, K. K., Ingram, S., Mudigere, D., Krishnamoorthi, R., Nair, K., Smelyanskiy, M., and Annavaram, M. {Check-N-Run}: A checkpointing system for training deep learning recommendation models. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 929–943, 2022.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Hu, Z., Zou, X., Xia, W., Jin, S., Tao, D., Liu, Y., Zhang, W., and Zhang, Z. Delta-dnn: Efficiently compressing deep neural networks via exploiting floats similarity. In *Proceedings of the 49th International Conference on Parallel Processing*, pp. 1–12, 2020.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Jin, H., Wu, D., Zhang, S., Zou, X., Jin, S., Tao, D., Liao, Q., and Xia, W. Design of a quantization-based dnn delta compression framework for model snapshots and federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 34(3):923–937, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Rao, K. R. and Hwang, J. J. *Techniques and standards for image, video, and audio coding*. Prentice-Hall, Inc., 1996.

Sashank, J. R., Satyen, K., and Sanjiv, K. On the convergence of adam and beyond. In *International conference on learning representations*, volume 5, 2018.

Shi, N. and Li, D. Rmsprop converges with proper hyperparameter. In *International conference on learning representation*, 2021.

Shu, H., Li, W., Tang, Y., Zhang, Y., Chen, Y., Li, H., Wang, Y., and Chen, X. Tinysam: Pushing the envelope for efficient segment anything model. *arXiv preprint arXiv:2312.13789*, 2023.

Sterne, J. *MP3: The meaning of a format*. Duke University Press, 2012.

Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., and Xu, C. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33:10936–10947, 2020.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, Y., Chen, H., Tang, Y., Guo, T., Han, K., Nie, Y., Wang, X., Hu, H., Bai, Z., Wang, Y., et al. Pangu-π: Enhancing language model architectures via nonlinearity compensation.

Wang, Y., Chen, H., Tang, Y., Guo, T., Han, K., Nie, Y., Wang, X., Hu, H., Bai, Z., Wang, Y., et al. Pangu-π: Enhancing language model architectures via nonlinearity compensation. *arXiv preprint arXiv:2312.17276*, 2023.

Wu, X., Zeng, F., Wang, X., Wang, Y., and Chen, X. Ppt: Token pruning and pooling for efficient vision transformers. *arXiv preprint arXiv:2310.01812*, 2023.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

Zhang, Y., Chen, C., Shi, N., Sun, R., and Luo, Z.-Q. Adam can converge without any modification on update rules. *Advances in neural information processing systems*, 35: 28386–28399, 2022.

A. Convergence Analysis.

Theorem A.1. *According the convergence analysis in Adam (Kingma & Ba, 2014), assume that the function* f_t *has bounded gradients,* $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ *for all* $\theta \in R^d$ *and distance between any* θ_t *generated by Adam is bounded,* $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ *for any* $m, n \in \{1, \ldots, T\}$, *and* $\beta_1, \beta_2 \in [0, 1)$ *satisfy* $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. *Let* $\alpha_t = \frac{\alpha}{\sqrt{t}}$ *and* $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. *If we prune the moments with a mask* $\mathcal{M}_o \in \{0, 1\}$ *at iteration* τ, *Adam could also achieves the following guarantee, for all* $T \geq 1$.

$$\begin{aligned}\tilde{R}(T) \leq & \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \sqrt{T \widehat{v}_{T,i}} + \frac{\alpha\left(1+\beta_1\right) G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha} \sum_{i=1}^{d} \sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})} \sqrt{t} \\ & + \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \left(\sqrt{T \widehat{v}_{\tau,i}} - \sqrt{T \widehat{v}_{\tau,i} \mathcal{M}_o}\right)\end{aligned} \tag{11}$$

Proof. According the convergence analysis in Adam (Kingma & Ba, 2014), assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Adam achieves the following guarantee, for all $T \geq 1$.

$$R(T) \leq \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \sqrt{T \widehat{v}_{T,i}} + \frac{\alpha\left(1+\beta_1\right) G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha\beta_1(1-\lambda)^2}, \tag{12}$$

where $R(T)$ is the regret:

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \tag{13}$$

This theorem could be obtained by the following:

$$\begin{aligned}R(T) \leq & \sum_{i=1}^{d} \frac{1}{2\alpha_1\left(1-\beta_1\right)} \left(\theta_{1,i} - \theta_{,i}^*\right)^2 \sqrt{\widehat{v}_{1,i}} + \sum_{i=1}^{d} \sum_{t=2}^{T} \frac{1}{2\left(1-\beta_1\right)} \left(\theta_{t,i} - \theta_{,i}^*\right)^2 \left(\frac{\sqrt{\widehat{v}_{t,i}}}{\alpha_t} - \frac{\sqrt{\widehat{v}_{t-1,i}}}{\alpha_{t-1}}\right) \\ & + \frac{\beta_1 \alpha G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \frac{\alpha G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \\ & + \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\beta_{1,t}}{2\alpha_t\left(1-\beta_{1,t}\right)} \left(\theta_{,i}^* - \theta_{t,i}\right)^2 \sqrt{\widehat{v}_{t,i}}\end{aligned} \tag{14}$$

$$\begin{aligned}R(T) \leq & \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \sqrt{T \widehat{v}_{T,i}} + \frac{\alpha\left(1+\beta_1\right) G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \frac{D_\infty^2}{2\alpha} \sum_{i=1}^{d} \sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})} \sqrt{t \widehat{v}_{t,i}} \\ \leq & \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \sqrt{T \widehat{v}_{T,i}} + \frac{\alpha\left(1+\beta_1\right) G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \\ & + \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha} \sum_{i=1}^{d} \sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})} \sqrt{t}\end{aligned} \tag{15}$$

In our method, we prune some variables for the momentum, *i.e.*, a mask $\mathcal{M}_o$ is applied for m and v. Assume that we prune the momentum at iteration τ, the convergence is the same as original optimization process for iteration 1 to iteration τ. However, the first and second moment vectors v_τ and m_τ become $v_\tau \mathcal{M}_o$ and $m_\tau \mathcal{M}_o$ at iteration τ.

From Eq. 14, we have:

$$\begin{aligned}\tilde{R}(T) \leq & \sum_{i=1}^{d} \frac{1}{2\alpha_1\left(1-\beta_1\right)}\left(\theta_{1,i}-\theta_{,i}^{*}\right)^2 \sqrt{\widehat{v}_{1,i}}+\sum_{i=1}^{d}\sum_{t=2}^{\tau-1} \frac{1}{2\left(1-\beta_1\right)}\left(\theta_{t,i}-\theta_{,i}^{*}\right)^2\left(\frac{\sqrt{\widehat{v}_{t,i}}}{\alpha_t}-\frac{\sqrt{\widehat{v}_{t-1,i}}}{\alpha_{t-1}}\right) \\ & \sum_{i=1}^{d} \frac{1}{2\left(1-\beta_1\right)}\left(\theta_{\tau,i}-\theta_{,i}^{*}\right)^2\left(\frac{\sqrt{\mathcal{M}_o \widehat{v}_{\tau,i}}}{\alpha_\tau}-\frac{\sqrt{\widehat{v}_{\tau-1,i}}}{\alpha_{\tau-1}}\right)+\sum_{i=1}^{d}\sum_{t=\tau+1}^{T} \frac{1}{2\left(1-\beta_1\right)}\left(\theta_{t,i}-\theta_{,i}^{*}\right)^2\left(\frac{\sqrt{\widehat{v}_{t,i}}}{\alpha_t}-\frac{\sqrt{\widehat{v}_{t-1,i}}}{\alpha_{t-1}}\right) \\ & +\frac{\beta_1 \alpha G_{\infty}}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2+\frac{\alpha G_{\infty}}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2 \\ & +\sum_{i=1}^{d}\sum_{t=1}^{T} \frac{\beta_{1,t}}{2\alpha_t\left(1-\beta_{1,t}\right)}\left(\theta_{,i}^{*}-\theta_{t,i}\right)^2 \sqrt{\widehat{v}_{t,i}}\end{aligned} \tag{16}$$

Under similar assumption as Eq. 17, we could have the following regret bound:

$$\begin{aligned}\tilde{R}(T) \leq & \frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d} \sqrt{T \widehat{v}_{T,i}}+\frac{\alpha\left(1+\beta_1\right) G_{\infty}}{\left(1-\beta_1\right)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d}\left\|g_{1:T,i}\right\|_2+\frac{D_{\infty}^2 G_{\infty}\sqrt{1-\beta_2}}{2\alpha} \sum_{i=1}^{d}\sum_{t=1}^{t} \frac{\beta_{1,t}}{\left(1-\beta_{1,t}\right)}\sqrt{t} \\ & +\frac{D^2}{2\alpha\left(1-\beta_1\right)} \sum_{i=1}^{d}\left(\sqrt{T \widehat{v}_{\tau,i}}-\sqrt{T \widehat{v}_{\tau,i}\mathcal{M}_o}\right)\end{aligned} \tag{17}$$

□

B. More Visualization of Compressed LLM.

Here we show more visualization results in Figure 6 and Figure 7.

Question: What is the area of a circle with a radius of two meters?

Original PanGu-π-1B: The formula for the area of a circle is A = πr^2, where r is the radius.
Substituting r = 2 meters, we get:
$A = \pi(2)^2$
$A = 4\pi$ square meters
Therefore, the area of a circle with a radius of 2 meters is approximately 4π square meters or approximately 12.57 square meters.

Compressed PanGu-π-1B: The formula for the area of a circle is $A = \pi r^2$, where r is the radius. To find the area of a circle with a radius of two meters, we can use the formula:
$A = \pi(2)^2$
$A = \pi(4)$
$A \approx 12.57$ square meters
So, the area of a circle with a radius of two meters is approximately 12.57 square meters.

Question: What are the seven continents on Earth?

Original PanGu-π-1B: The seven continents are: 1. Asia 2. Africa 3. North America 4. South America 5. Antarctica 6. Europe 7. Australia/Oceania

Compressed PanGu-π-1B: The seven continents are: 1. Asia 2. Africa 3. North America 4. South America 5. Antarctica 6. Europe 7. Australia

Figure 6. Q&A example to show the difference between our compressed model and the original model.

Question: How to find a suitable picnic spot?

Original PanGu-π-1B: Finding a picnic spot can be a great way to unwind and enjoy a meal with friends or family. Here are a few tips to help you find the perfect picnic spot: 1. Look for an outdoor space: Whether it's a park, a beach, or a forest, finding an outdoor space to set out a picnic can be a great way to enjoy the outdoors. 2. Check for shady areas: A picnic should be a fun and enjoyable experience, so make sure to check for shady areas where people might be hiding or eating. 3. Look for a busy area: If you're picnicking during the day, look for an area with a lot of people gathered, like a park or mall park. 4. Check for views: A great picnic spot should have a great view, so think of a location that overlooks a large body of water or a park entrance. 5. Look for a comfortable spot: A picnic should be a relaxing experience, so think of a location that is comfortable and easy to lie down or sleep in. 6. Take into account seasonal weather: If you're picnicking in the spring or fall, look for a location that is shaded during those months, as those times can be pleasant. 7. Ask locals: If you're trying to find a picnic spot, don't be afraid to ask locals. They may have some of the best spots for picnicking and enjoying a good picnic.

Compressed PanGu-π-1B: There are several ways to find a suitable picnic spot: 1. Ask for recommendations: Ask family and friends for recommendations on a suitable spot for the picnic. They may know a place that they have used or that they think is a good choice. 2. Check online: Check online for suggestions on picnic spots in your area. Many websites have a section for finding suitable spots for picnics, or you can search for them by using search engines. 3. Look for public spaces: Look for parks, gardens, or other public spaces that are often free of trees and open spaces for picnics. 4. Check with local authorities: Contact your local government to see if they have any upcoming events or festivals that require a picnic. They may be able to provide you with some recommendations on a suitable spot. 5. Ask for a description: Ask the host or hostess at the restaurant to provide a description of the picnic spot, including the number of people, the area, and the amenities available. Remember to take necessary precautions to ensure the safety of yourself and others while enjoying a picnic.

Question: How to take beautiful landscape photos?

Original PanGu-π-1B: 1. Understand your subject: landscape photos generally require some understanding of the subject in order to accurately capture its beauty. Study the subject, including its culture, history, and natural environment, to be able to anticipate its behavior and lighting conditions. 2. Use the right lighting: Landscape photography often requires the use of artificial lighting to illuminate the subject. Experiment with different lighting techniques, such as natural lighting, studio lighting, or table lamp lighting, to create the desired effect. 3. Adjust camera settings: Most cameras have settings that can either adjust depth of field, aperture, and ISO. Experiment with these settings to find the right balance of light and depth, and to get the desired exposure ratio, depth, and color. 4. Take multiple shots: Taking multiple photos from different angles can help to create a more dynamic and visually interesting landscape. Take advantage of vantage points, such as bridges, buildings, and mountains, to increase the subject's exposure and clarity. 5. Add foreground elements: Adding a foreground element, such as trees, mountains, or buildings, can help to add depth and context to your photo. Use the rule of thirds to create a pleasing balance between the foreground and background. 6. Edit your photos: Once you've captured your landscape photos, take the time to edit your photos. You can adjust brightness, contrast, and saturation, and add filters or editing tools to enhance the subject. 7. Practice, practice, practice: The more you practice taking landscape photos, the better you'll get at capturing the beauty of the landscape. Keep experimenting with different lighting, camera settings, and editing techniques to keep your photos looking their best.

Compressed PanGu-π-1B: 1. Understand your environment: Be sure to study your chosen location, the time and weather, and the type of subject you are photographing. 2. Choose the right camera: Not all cameras are created equal, so take the time to research and choose the camera that best suits your needs. 3. Get the right lighting: The lighting can dramatically affect the mood and tone of your photos. Try to capture the light sources that are most effective for your subject. 4. Experiment with composition: Composition is an essential part of photography, so experiment with different angles, vantage points, and compositions to see what works best for your subject. 5. Edit your photos: Once you have your compelling landscape photos, it's time to edit them. You can use software to add filters, adjust brightness and contrast, and enhance the overall look of your photos. 6. Share your photos: Take a camera and post your photos on social media platforms like Instagram, Facebook, and YouTube. This will allow others to view and appreciate your work, and also help you gain more followers.

Figure 7. Q&A example to show the difference between our compressed model and the original model.