Delta-CoMe: Training-Free Delta-Compression with Mixed-Precision for Large Language Models

Bowen Ping[1*] **Shuo Wang**[2*] **Hanqing Wang**[3] **Xu Han**[2,4,5] **Yuzhuang Xu**[2] **Yukun Yan**[2]
Yun Chen[3] **Baobao Chang**[1] **Zhiyuan Liu**[2,4,5†] **Maosong Sun**[2,4,5†]
[1]Peking University [2]Dept. of Comp. Sci. & Tech., Tsinghua University, Beijing, China
[3]Shanghai University of Finance and Economics
[4]Institute for AI, Tsinghua University, Beijing, China
[5]Beijing National Research Center for Information Science and Technology

Abstract

Fine-tuning is a crucial process for adapting large language models (LLMs) to diverse applications. In certain scenarios, such as multi-tenant serving, deploying multiple LLMs becomes necessary to meet complex demands. Recent studies suggest decomposing a fine-tuned LLM into a base model and corresponding delta weights, which are then compressed using low-rank or low-bit approaches to reduce costs. In this work, we observe that existing low-rank and low-bit compression methods can significantly harm the model performance for task-specific fine-tuned LLMs (e.g., WizardMath for math problems). Motivated by the long-tail distribution of singular values in the delta weights, we propose a delta quantization approach using mixed-precision. This method employs higher-bit representation for singular vectors corresponding to larger singular values. We evaluate our approach on various fine-tuned LLMs, including math LLMs, code LLMs, chat LLMs, and even VLMs. Experimental results demonstrate that our approach performs comparably to full fine-tuned LLMs, surpassing both low-rank and low-bit baselines by a considerable margin. Additionally, we show that our method is compatible with various backbone LLMs, such as Llama-2, Llama-3, and Mistral, highlighting its generalizability. [3]

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Jiang et al., 2023) are increasingly becoming the standard for a wide range of downstream tasks (Luo et al., 2023a; Yu et al., 2023; Wei et al., 2023; Luo et al., 2023b; Liu et al., 2024a; Wang et al., 2023), significantly surpassing conventional small models. To meet the demands of various application domains and scenarios, many researchers direct their attention to developing advanced alignment or adaptation algorithms together with diverse training data to learn aligned LLMs based on generally pre-trained models. For instance, Luo et al. (2023a) propose a reinforcement learning from evol-instruct feedback (RLEIF) method to construct LLMs with strong mathematical reasoning abilities. Similarly, Yu et al. (2023) employ a bootstrapping method to diversify mathematical questions and then fine-tune open-source LLMs to build mathematical models. For code generation, Luo et al. (2023b) adapt the evol-instruct method to the coding domain, resulting in the WIZARDCODER model, which demonstrates superior coding abilities compared to generally trained LLMs. Additionally, Wei et al. (2023) enhance the capabilities of open-source code LLMs by using automatically generated high-quality instruction data based on

[*] Equal contribution.
[†] Corresponding authors.
[3] Code will be publicly available at `https://github.com/thunlp/Delta-CoMe`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-0978

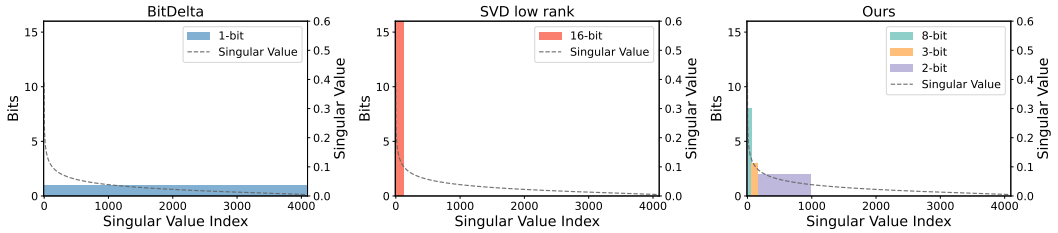


Figure 1: **Left**: illustration of BitDelta (Liu et al., 2024b), which employs 1-bit quantization for all the delta weights. **Middle**: illustration of low-rank compression (Ryu et al., 2023b), retaining the top-k singular values and the corresponding singular vectors. **Right**: illustration of the proposed Delta-CoMe method, which represents the singular vectors of larger singular values using high-bit vectors while compressing the singular vectors of smaller singular values into low-bit representations. This method is inspired by the long-tail distribution of singular values in delta weights.

existing code snippets. Wang et al. (2023) utilize various resources of mixed quality and design a new conditioned reinforcement learning fine-tuning method to train the OPENCHAT model. Beyond the text modality, some studies propose fine-tuning pre-trained LLMs to understand other modalities. For instance, Liu et al. (2024a) construct a multi-modal instruction tuning dataset and develop the LLAVA model, which can understand both text and images.

Building on the aforementioned alignment approaches, LLMs are endowed with specialized capabilities that align with distinct user demands and application requirements (Liu et al., 2024b). In certain scenarios, deploying multiple LLMs with different abilities is necessary. For example, in multi-tenant serving, different LLMs may be needed to satisfy various users. Additionally, some complex tasks consist of multiple sub-tasks, each requiring different model capabilities. To address these tasks, we should organize and deploy a group of LLMs simultaneously. A straightforward question arises: why not use a single general LLM that encompasses all the necessary capabilities? For example, we could develop one model that can both understand images and generate code programs. To our knowledge, LLMs with various capabilities (e.g., GPT-4[4]) typically have an enormous number of parameters, making them impractical for resource-limited situations (e.g., edge-side scenarios).

In pursuit of this objective, a field of research advocates for the minimization of expenses associated with multi-model serving. Delta-compression emerges as a crucial and viable approach in this context, offering the potential to decrease both storage requirements and GPU memory utilization in scenarios involving multiple models. The primary objective of delta-compression is to minimize the size of the delta weights between aligned and pre-trained LLMs (e.g., LLAMA-2-CHAT and LLAMA-2). Ryu et al. (2023b) identify the low-rank nature of delta weights and enhance storage efficiency through low-rank approximation. Alternatively, Liu et al. (2024b) propose a 1-bit quantization approach, termed BitDelta, to further reduce the size of delta weights. They validate the effectiveness of BitDelta across various chat models, including LLAMA-2-CHAT (Touvron et al., 2023), VICUNA[5], and WIZARDLM (Xu et al., 2023). In this work, we reassess the performance of both low-rank and low-bit delta-compression methods across a diverse range of aligned LLMs, encompassing mathematical, coding, chat, and multi-modal LLMs. Our experimental results (e.g., Table 3) reveal that current low-rank and low-bit compression techniques may significantly degrade the performance of aligned LLMs. These results motivate us to explore more advanced delta-compression methods capable of achieving performance nearly equivalent to the aligned LLMs before compression.

Inspired by the long-tail distribution of singular values, as illustrated in Figure 1, we propose allocating higher-bit representations for singular vectors associated with larger singular values, given their greater impact on the approximation of delta weights prior to compression. Conversely, for singular vectors associated with smaller singular values, we employ low-bit formats to reduce the delta size. For singular values that are extremely small, we omit the corresponding singular vectors altogether. The resulting method, which we term Delta-CoMe, can be viewed as a hybrid of low-rank and low-bit compression techniques. Delta-CoMe outperforms both the low-rank compression method and BitDelta. Moreover, our method achieves performance comparable to that of the full

[4]https://chatgpt.com

[5]https://lmsys.org/blog/2023-03-30-vicuna

https://doi.org/10.52202/079017-0978

aligned LLMs. For instance, Delta-CoMe attains an average score of 53.2 across eight representative tasks, closely matching the average score of 53.5 achieved by the aligned LLMs. In comparison, the scores of the low-rank and low-bit baselines are 47.8 and 49.3, respectively.

Further, we compare the performance of the involved delta-compression methods to LoRA (Hu et al., 2022), a widely-used delta-tuning approach (Wang et al., 2024). The primary distinction between delta-compression and delta-tuning is that delta-compression first optimizes the full model and then converts the modified weights into a lightweight module, reducing inference costs in multi-model settings. In contrast, delta-tuning primarily aims to lower training costs. Our experimental results demonstrate that the proposed Delta-CoMe method significantly outperforms LoRA, with scores of 41.9 versus 29.8, respectively. These results suggest that delta-compression can deliver superior performance in multi-model settings compared to delta-tuning.

Finally, Delta-CoMe can achieve more than $10\times$ GPU memory and disk storage savings, enabling the deployment of multiple models with limited resources. For practical application, we implement a Triton (Tillet et al., 2019) kernel tailed for Delta-Come, achieving approximately a $3\times$ speedup compared to the PyTorch implementation.

Our contribution can summarized as follows:

- We propose a mixed-precision delta-compression method that employs varying bit-widths for different singular vectors based on their singular values;
- We validate the effectiveness of the proposed method across different types of aligned LLMs of varying sizes, including mathematical, coding, chat, and multi-modal LLMs;
- We conduct in-depth analyses to understand the superior performance of our method over low-rank and low-bit baselines. Our method can also outperform delta-tuning approaches such as LoRA, demonstrating that the proposed delta-compression method is more practical for multi-model serving scenarios.
- We verify that the proposed method can achieve over $10\times$ saving in GPU memory and disk storage. By constructing a Triton kernel, we can achieve approximately a $3\times$ speedup, demonstrating the hardware compatibility of Delta-CoMe.

2 Related Work

2.1 Delta-Compression

Recently, delta-compression has garnered increasing interest in the LLM community due to its ability to substantially diminish the storage and inference expenses associated with serving multiple models. GPT-Zip extends the GPTQ approach (Frantar et al., 2023) to compress the delta weights between aligned models and the backbone model, successfully using 2-bit delta weights to approximate the model. Additionally, they sparsify the quantized delta weights to further reduce storage costs. However, the sparsification technique can hardly reduce GPU memory usage during inference. Similarly, Yu et al. (2024) find that dropping the majority of the delta weights has a limited effect on the performance of aligned LLMs. Ryu et al. (2023a) identify the low-rank property of delta weights and propose reducing the storage requirements of aligned LLMs through low-rank approximation. Yao & Klimovic (2023) adopt the concept of delta-compression to develop a multi-tenant serving system, DeltaZip. Most recently, Liu et al. (2024b) introduced BitDelta, which successfully quantizes the delta weights into 1-bit. However, they only examined the performance of this compression method using chat LLMs, leaving a wide range of other types of aligned LLMs unexplored. In this work, we propose leveraging the benefits of both low-rank and low-bit compression methods by using varying bit-widths to represent different components of the delta weights. We evaluate representative low-rank and low-bit delta-compression methods across various types of aligned LLMs to provide a comprehensive comparison of these methods.

2.2 Model Compression with Mix-Precision

Using mixed-precision to compress the model weights is an effective technique that has been investigated in many previous studies. SpQR (Dettmers et al., 2023) isolates a small number of outlier weights and retains them with high-precision, while keeping the other weights at low-precision,

https://doi.org/10.52202/079017-0978

resulting in a significant performance improvement. Based on activations, Agile-Quant (Shen et al., 2024) utilizes token pruning to achieve mixed-precision quantization of both weights and activations. Bablani et al. (2023) propose employing varying bit-widths for different layers of the model, while Yao et al. (2021) propose quantizing activations and model weights with different precisions. In this work, we propose using mixed-precision compression for different singular vectors of the delta model, marking the first method to introduce mixed-precision compression for delta weights.

3 Approach

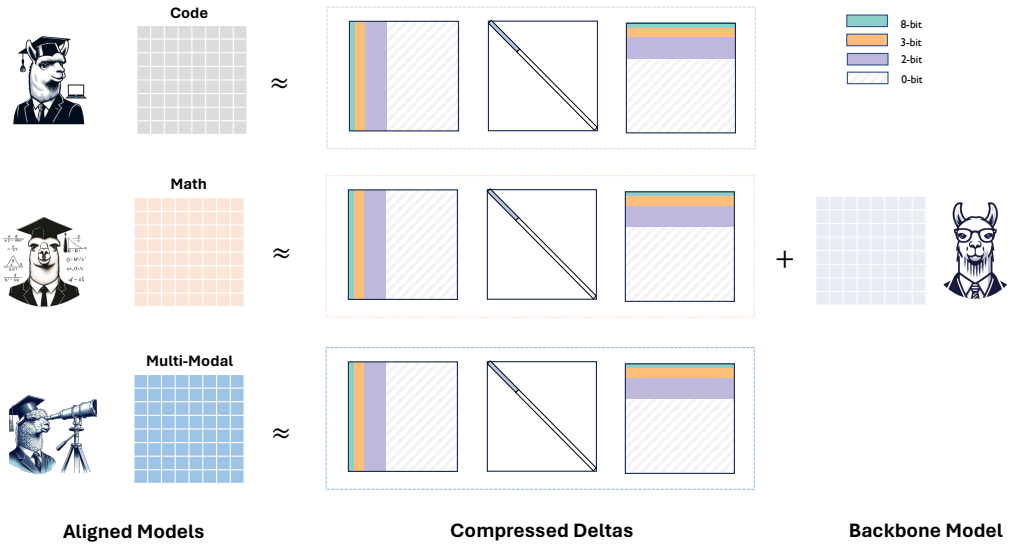


Figure 2: Illustration of Delta-CoMe, where we utilize varying bit-widths for singular vectors with different singular values. Singular vectors corresponding to larger singular values are assigned higher bit-widths. For extremely small singular values, we omit the singular vectors (i.e., 0-bit).

3.1 Preliminaries

For a **backbone LLM** $\boldsymbol{\theta}_b$, we can customize it into an **aligned model** $\boldsymbol{\theta}_a$ for a specific purpose using advanced alignment algorithms (Xu et al., 2023; Luo et al., 2023a; Yu et al., 2023; Luo et al., 2023b; Wei et al., 2023; Liu et al., 2024a). In some practical scenarios, as mentioned in Section 1, we may need to deploy multiple LLMs at the same time. Formally, we should store and deploy a series of aligned LLMs $\left\{\boldsymbol{\theta}_a^{(1)}, \cdots, \boldsymbol{\theta}_a^{(N)}\right\}$, where N is the number of aligned models. The total size of the group of aligned models is $N \times M$, where M is the size of one model. We use $\boldsymbol{\Delta}$ to represent the delta weights between the aligned model and the backbone model, which is given by

$$\boldsymbol{\Delta}^{(n)} = \boldsymbol{\theta}_a^{(n)} - \boldsymbol{\theta}_b, \tag{1}$$

where $\boldsymbol{\theta}^{(n)}$ is the n-th aligned LLM. Note that the sizes of $\boldsymbol{\Delta}^{(n)}$, $\boldsymbol{\theta}_a^{(n)}$, and $\boldsymbol{\theta}_b$ are the same.

Delta-compression aims to compress the delta weights $\boldsymbol{\Delta}^{(n)}$ into $\hat{\boldsymbol{\Delta}}^{(n)}$, where the latter has significantly fewer parameters. After delta-compression, we can only maintain one backbone model and N compressed delta models: $\left\{\boldsymbol{\theta}_b, \hat{\boldsymbol{\Delta}}^{(1)}, \cdots, \hat{\boldsymbol{\Delta}}^{(N)}\right\}$. The total size is decreased from $N \times M$ to $(1 + \alpha N) \times M$, where α is the compression ratio. During inference, we can restore each aligned LLM in the following way:

$$\hat{\boldsymbol{\theta}}_a^{(n)} = \boldsymbol{\theta}_b + \hat{\boldsymbol{\Delta}}^{(n)}. \tag{2}$$

For a good delta-compression method, we expect it can achieve a smaller α, while making $\hat{\boldsymbol{\theta}}_a^{(n)}$ attain comparable performance with $\boldsymbol{\theta}_a^{(n)}$. BitDelta (Liu et al., 2024b), to our knowledge, is the most recent study that successfully quantizes delta weights into 1-bit, which means that $\alpha = 1/16$ when the

original aligned model is represented by FP16 or BF16. In this work, we propose to improve the performance of delta-compression methods by inducing mixed-precision quantization, which will be detailed in the following sub-sections.

3.2 Delta Decomposition

Previous works have investigated mixed-precision model compression methods at the token (Shen et al., 2024) or layer level (Bablani et al., 2023). For delta-compression, we propose employing mixed-precision for different singular vectors. We first use the SVD algorithm to decompose each delta matrix:

$$\Delta\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{3}$$

where $\Delta\mathbf{W} \in \mathbb{R}^{h_{\text{out}} \times h_{\text{in}}}$, $\mathbf{U} \in \mathbb{R}^{h_{\text{out}} \times h_{\text{out}}}$, $\mathbf{\Sigma} \in \mathbb{R}^{h_{\text{out}} \times h_{\text{in}}}$, $\mathbf{V} \in \mathbb{R}^{h_{\text{in}} \times h_{\text{in}}}$. Intuitively, the singular vectors associated with larger singular values have a greater impact on the approximation of the delta matrix $\Delta\mathbf{W}$, we thus spend more bits for these vectors to reduce the quantization error.

3.3 Mixed-Precision Quantization

Some representative quantization methods, such as GPTQ (Frantar et al., 2023), aims to minimize the following objective:

$$\hat{\mathbf{W}} = \text{Quant}_k(\mathbf{W}, \mathbf{X}) = \underset{\hat{\mathbf{W}}}{\text{argmin}} ||\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}||^2, \tag{4}$$

where $\mathbf{X} \in \mathbb{R}^{h_{\text{in}}}$ is the input to the parameter $\mathbf{W}$ and $\hat{\mathbf{W}}$ is the corresponding quantized parameter. We use Quant_k to denote the k-bit quantization algorithm. In this work, we employ the widely-used GPTQ method with group_size $= 128$ for cases where $k > 1$, and BitDelta for 1-bit quantization. For a certain group of singular vectors, let r_{begin} and r_{end} represent the start and end indices, respectively. The quantization of the singular vectors can be given by

$$\begin{aligned}
&\hat{\mathbf{V}}[:, r_{\text{begin}} : r_{\text{end}}]^\top = \text{Quant}_k(\mathbf{V}[:, r_{\text{begin}} : r_{\text{end}}]^\top, \mathbf{X}), \\
&\hat{\mathbf{U}}[:, r_{\text{begin}} : r_{\text{end}}] = \\
&\qquad \text{Quant}_k(\mathbf{U}[:, r_{\text{begin}} : r_{\text{end}}], \mathbf{\Sigma}[r_{\text{begin}} : r_{\text{end}}, r_{\text{begin}} : r_{\text{end}}]\hat{\mathbf{V}}[:, r_{\text{begin}} : r_{\text{end}}]^\top \mathbf{X}).
\end{aligned} \tag{5}$$

As illustrated in Figure 2, we use varying quantization bits for different groups of singular vectors. By employing different mixed-precision strategies, we can control the trade-off between achieving a small delta size and maintaining high performance. We will provide more details about the exploration of the mixing strategy in Section 5.1.

4 Experimental Setup

To thoroughly investigate the proposed delta-compression method Delta-CoMe and the involved baselines, we examine the performance of different methods across several tasks, which are typical applications of recent aligned LLMs.

4.1 Tasks

Mathematical Problem Solving Solving mathematical problems is a challenging task for modern LLMs. For this task, we employ GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) as the evaluation datasets, which are among the most popular mathematical benchmarks for LLMs. The reported score is accuracy, which is estimated by comparing the ground-truth number with the result calculated by the model.

Code Generation The ability to process code is crucial for numerous practical applications, including data analysis and LLM-based agents. For this task, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) as the evaluation datasets, which are widely used in recent studies. The reported score is the pass rate, indicating that the model-generated code can successfully run the test cases in one pass (i.e., pass@1).

https://doi.org/10.52202/079017-0978

Chat The chat ability enables LLMs to interact with users, providing helpful and safe suggestions or responses based on the user's requests. A good chat model is expected to be well aligned with human preferences. For evaluating chat LLMs, we select TruthfulQA (Lin et al., 2022) and SafetyBench (Zhang et al., 2023) as the evaluation datasets, which measure helpfulness and safety, respectively. The reported score is the accuracy, indicating that the choice of the model is correct.

Multi-Modal Chat Vision-language models (VLMs) are attracting increasing attention due to their ability to process both text and images. Most recent VLMs are based on pre-trained visual encoders and language models, with the language models fine-tuned to understand the visual signal. For this task, we use GQA (Hudson & Manning, 2019) and TextVQA (Singh et al., 2019). The reported score is the accuracy, indicating that the choice of the model is correct.

4.2 Models

Table 1: Selected backbone and aligned models for the examined four tasks.

Task	7B Models		13B Models	
	Backbone	**Aligned**	**Backbone**	**Aligned**
Math	LLAMA-2	WIZARDMATH-V1.0	LLAMA-2	WIZARDMATH-V1.0
Code	CODELLAMA-PY	MAGICODERS-CL	CODELLAMA-PY	WIZARDCODER-PY-V1.0
Chat	LLAMA-2	LLAMA-2-CHAT	LLAMA-2	LLAMA-2-CHAT
Multi-Modal	VICUNA-V1.5	LLAVA-V1.5	VICUNA-V1.5	LLAVA-V1.5

For the four tasks, we provide the backbone and aligned models in Table 1. All the model weights are open-sourced by the authors. We use both 7B and 13B models to make a thorough comparison between different delta-compression models. During inference, we use greedy search.

4.3 Baselines

We employ two representative baselines, including SVD-based low-rank compression and BitDelta (Liu et al., 2024b). For the low-rank baseline, we re-implement the method, while for BitDelta, we use the code open-sourced by the authors.[6] All methods are evaluated on NVIDIA A100 GPUs.

5 Experimental Results

5.1 Exploration of Mixed-Precision Strategies

To determine which bit-width to use and how many singular vectors to quantize, we conduct a preliminary experiment using different mixed-precision strategies. We examine three types of strategies: single-precision, double-precision, and triple-precision settings. The size of the compressed delta remains consistent across all settings. For single-precision compression, we set r_{begin} to 0, and r_{end} is set to guarantee that the delta size is the same as BitDelta (Liu et al., 2024b). In other words, the compression ratio α for all settings is 1/16. Formally, for a delta matrix $\Delta\mathbf{W} \in \mathbb{R}^{h_{\text{out}} \times h_{\text{in}}}$, r_{begin} and r_{end} are set to satisfy the following equation:

$$k \times (r_{\text{end}} - r_{\text{begin}})(h_{\text{out}} + h_{\text{in}}) = 16 \times \alpha h_{\text{out}} h_{\text{in}}, \quad (6)$$

where α is set to 1/16 in our experiments, which is the same as BitDelta. In double-precision settings, r_{begin} and r_{end} are set to 0 and 2, respectively, for the first precision. For the second precision, r_{begin}

Table 2: Comparison of different mixed-precision strategies.

# Precision	Setting	GSM8K
Single	1	45.6
	2	50.6
	3	**51.8**
	4	51.6
	8	47.8
	16	43.3
Double	16 + 3	52.5
	8 + 3	**53.1**
	4 + 3	52.2
	3 + 2	52.3
Triple	16 + 8 + 3	53.2
	8 + 4 + 3	52.2
	8 + 3 + 2	**53.6**

[6] https://github.com/FasterDecoding/BitDelta.

is set to 2, and r_{end} is adjusted so that the total delta size is 1/16 of the uncompressed delta. In triple-precision settings, r_{begin} and r_{end} are set to 0 and 2, respectively, for the first precision. r_{begin} and r_{end} are set to 2 and 34, respectively, for the second precision. For the third precision, r_{begin} is set to 34, and r_{end} is adjusted so that the total delta size is 1/16 of the uncompressed delta. Since the diagonal matrix $\mathbf{\Sigma}$ occupies little storage, the averaged bit-width for triple-precision compression is approximately

$$\frac{h_{\text{out}} + h_{\text{in}}}{h_{\text{out}} h_{\text{in}}} \sum_{i=1}^{3} k^{(i)} (r_{\text{end}}^{(i)} - r_{\text{begin}}^{(i)}). \tag{7}$$

We conduct experiments on the math task, and the results are shown in Table 2. We find that the 3-bit setting performs best among the single-precision settings. Therefore, we keep the 3-bit setting and add other bit-widths to form double-precision settings. Among the double-precision settings, "8+3" achieves the highest score, which is then combined with an additional bit-width to form triple-precision settings. We find that the best double-precision setting can outperform the best single-precision setting, and the best triple-precision setting achieves the highest score across all the examined settings. We use "8+3+2" as the default setting in the following experiments.

5.2 Main Results

Tables 3 and 4 show the performance of different delta-compression methods on 7B and 13B models, respectively. Across all tasks, Delta-CoMe outperforms both baselines. While BitDelta (Liu et al., 2024b) can achieve near lossless performance on chat models, it significantly degrades the performance of math and code LLMs, a phenomenon not investigated by Liu et al. (2024b). Surprisingly, our method achieves good performance in the delta-compression of VLMs. To our knowledge, we are the first to investigate delta-compression for VLMs.

Table 3: The performance of different delta-compression methods on 7B aligned models.

Method	α	WizardMath		MagicoderS-CL		Llama-2-chat		Llava-v1.5		**Ave.**
		GSM8K	**MATH**	**HumanEval**	**MBPP**	**SafetyBench**	**TruthfulQA**	**GQA**	**TextVQA**	
Backbone	1	11.0	2.9	38.4	47.6	41.7	38.9	n/a	n/a	n/a
Aligned	1	55.2	10.9	70.7	69.2	59.5	44.6	62.0	58.2	53.5
Low-Rank	1/16	43.2	8.0	56.7	65.7	55.4	42.5	57.7	53.3	47.8
BitDelta	1/16	45.6	8.6	57.3	65.9	59.3	41.1	59.7	56.9	49.3
Delta-CoMe	1/16	**53.6**	**10.3**	**67.1**	**67.9**	**59.8**	**47.0**	**61.7**	**58.5**	**53.2**

Table 4: The performance of different delta-compression methods on 13B aligned models.

Method	α	WizardMath		WizardCoder		Llama-2-chat		Llava-v1.5		**Ave.**
		GSM8K	**MATH**	**HumanEval**	**MBPP**	**SafetyBench**	**TruthfulQA**	**GQA**	**TextVQA**	
Backbone	1	17.8	3.9	43.3	49.0	55.0	37.3	n/a	n/a	n/a
Aligned	1	63.9	14.0	60.4	66.9	62.7	43.9	63.2	61.3	54.5
Low-Rank	1/16	54.2	9.4	53.0	66.9	62.3	43.7	60.2	58.3	51.0
BitDelta	1/16	54.8	10.6	51.8	64.2	62.6	41.6	60.9	60.3	50.9
Delta-CoMe	1/16	**58.9**	**12.8**	**57.9**	**67.2**	**62.9**	**44.1**	**63.1**	**61.2**	**53.5**

5.3 Results on More Backbone Models

To investigate the generalization abilities of the delta-compression methods, we conduct experiments on aligned models based on other representative backbone LLMs. For additional backbones, we utilize Mistral-7B-v0.1 (Jiang et al., 2023) and Llama-3-8B[7]. The corresponding aligned

[7] `https://huggingface.co/meta-llama/Meta-Llama-3-8B`.

https://doi.org/10.52202/079017-0978

Table 5: Results on other representative backbones. The backbone of OPENCHAT-3.5-0106 (Wang et al., 2023) is MISTRAL-7B-V0.1 (Jiang et al., 2023). Both MISTRAL-7B-V0.1 and LLAMA-3-8B are widely-used open-source LLMs.

Method	α	OPENCHAT-3.5-0106				LLAMA-3-8B-INSTRUCT				Ave.
		GSM8K	HumanEval	TruthfulQA	SafetyBench	GSM8K	HumanEval	TruthfulQA	SafetyBench	
Backbone	1	52.2	28.7	61.0	42.1	44.8	33.5	43.6	43.9	43.7
Aligned	1	77.1	73.2	78.4	61.0	78.5	61.6	68.2	51.6	68.7
Low-Rank	1/16	50.5	52.4	76.9	49.0	68.3	46.3	67.5	51.3	57.8
BitDelta	1/16	70.3	54.9	78.4	50.0	67.6	56.1	68.6	50.2	62.0
Delta-CoMe	1/16	**74.8**	**59.8**	**78.9**	**62.6**	**77.1**	**60.4**	**69.1**	**51.8**	**66.8**

models are OPENCHAT-3.5-0106 (Wang et al., 2023) and LLAMA-3-8B-INSTRUCT, respectively. As shown in Table 5, our proposed Delta-CoMe method maintains superior performance over the two baselines, demonstrating its generalization ability.

5.4 Delta-Compression vs. Delta-Tuning

A closely related area to delta-compression is delta-tuning. While delta-tuning primarily aims to reduce the training cost of LLMs, delta-compression focuses on reducing the storage and inference cost for multi-model serving. It remains unclear whether delta-compression outperforms delta-tuning when using the same delta size. To investigate this, we trained LoRA (Hu et al., 2022) modules for all model parameters to compare delta-compression with delta-tuning. We set the LoRA rank to 128 and the scale factor to 16, using a cosine warmup schedule with a warmup ratio of 0.04 and a peak learning rate of 1e-4. For each task, we trained the LoRA for 3 epochs. For mathematical LoRA, the training dataset is from Yu et al. (2023), which consists of 395K training examples. For code LoRA, the training set is from Wei et al. (2023), which contains 186K training examples. For a fair comparison, we fine-tune all model parameters using the same dataset as used for LoRA training. We then apply different delta-compression methods to both the fine-tuned mathematical and code LLMs.

Table 6: Comparison between LoRA and delta-compression methods.

Method	Math		Code		Ave.
	GSM8K	MATH	HumanEval	MBPP	
Backbone	11.0	2.9	10.5	17.7	10.5
Aligned	65.4	18.6	43.2	44.9	43.0
LoRA	58.3	11.4	17.6	31.8	29.8
Low-Rank	54.8	5.5	26.2	42.6	32.3
BitDelta	47.8	10.7	26.2	41.9	31.7
Delta-CoMe	**65.1**	**18.0**	**39.6**	**44.9**	**41.9**

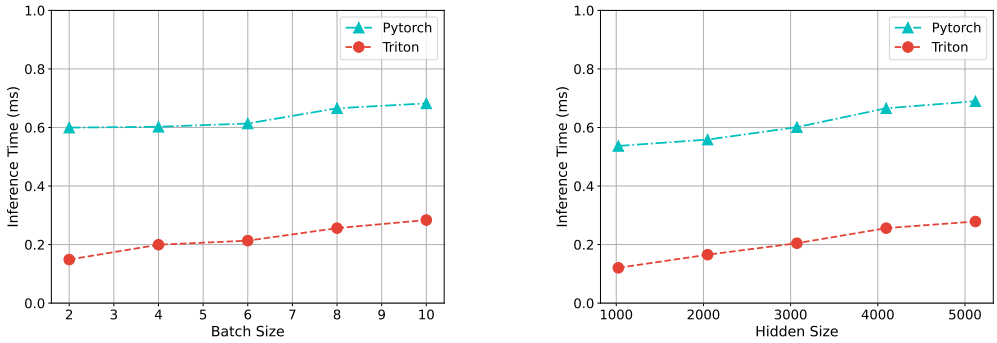
Table 6 shows the results of both delta-tuning and delta-compression methods. The results reveal that LoRA achieves superior performance compared to the low-rank compression approach and BitDelta in the mathematical task. However, when it comes to the coding task, LoRA exhibits lower performance than both low-rank compression and BitDelta. By contrast, our proposed delta-compression method (i.e., Delta-CoMe) consistently outperforms LoRA across all four benchmarks. Specifically, the performance of our method is close to that of the uncompressed aligned models (41.9 vs. 43.0), while the average score of LoRA is only 29.8. These results imply that learning an aligned model and then compressing it can achieve better results than delta-tuning.

5.5 Inference Speed and Memory Cost

For practical applications, we also examine the inference speed and memory cost of Delta-CoMe. In terms of inference speed, we implement a Triton kernel. Figure 3 shows the inference time of the PyTorch and Triton implementation of Delta-CoMe. Overall, we can achieve approximately a $3\times$ speedup across different settings. As Figure 3a shows, we first conduct an ablation experiment on varying batch sizes. Our implemented Triton kernel is consistently faster than the PyTorch implementation with different batch size settings. As Figure 3b depicts, we conduct an ablation experiment on hidden size to verify the adaptability of the Triton kernel to models of different sizes.

https://doi.org/10.52202/079017-0978

The Triton kernel can maintain a substantial speedup across different hidden sizes, demonstrating its ability to adapt to various models.



(a) Effect of batch size.

(b) Effect of hidden size.

Figure 3: Inference time of the PyTorch and Triton implementation of Delta-CoMe.

In Table 7, we show the GPU memory cost of deploying multiple aligned models that are fine-tuned from LLAMA-2-7B. The model parameters are represented in BF16 on a single 80G GPU. Without delta compression, a single GPU can not support 8 models, let alone more models. Using our proposed delta-compression method, we can load up to 50 models into one GPU, significantly reducing the deployment cost.

Table 7: GPU memory cost (GB).

Num. of Models	w/o DC	w/ DC
2	26.67	15.54
4	52.24	18.17
8	OOM	23.44
16	OOM	33.95
32	OOM	55.06
50	OOM	78.70

6 Analysis

6.1 Analysis of Quantization Error

To better understand the performance of various delta-compression methods, we estimate the quantization error as defined in Eq. (4). It is important to note that the error we calculate differs from that of GPTQ. Specifically, we use the mean square error between the activations of the uncompressed aligned model and those of the combination of the backbone model and the compressed delta model. The error is estimated on the GSM8K test set using WIZARDMATH-7B-V1.0 as the aligned model and LLAMA-2-7B as the backbone model. Since different layers have varying impacts on the final output (Wu et al., 2023), we distinguish low-, medium-, and high-layers when estimating the average quantization error. Specifically, the first 11 layers are designated as low-layers, the 12th to 22nd layers as medium-layers, and the last 10 layers as high-layers. Moreover, as outliers play a critical role in model compression (Dettmers et al., 2023; Lin et al., 2023), we also calculate the average error on outlier parameters. For each delta matrix $\Delta\mathbf{W}$, we select the top 1% of columns with the largest absolute values as outliers. Table 8 presents the results. We find that the average error of our methods (i.e., "Single" and "Triple") is substantially lower than both the low-rank baseline and BitDelta. Furthermore, the error of "Triple" is consistently less than that of "Single," reaffirming the necessity of mixed-precision compression for delta weights.

6.2 Case Study

We also present a detailed case study in Figure 4. Three delta-compression methods are examined: BitDelta, single-precision compression, and triple-precision compression. The reference answer is "104 hours". We observe that BitDelta makes mistakes initially, while single-precision compression generates an incorrect intermediate result at the second reasoning step. In contrast, our mixed-precision delta-compression method calculates the correct final answer.

https://doi.org/10.52202/079017-0978

Table 8: Approximation errors ($\times 10^{-2}$) at the activation level for different model parameters. "Low", "Medium", "High" represent low-, medium-, and high-layers, respectively. "All" means the error averaged across all the parameters, while "Out." denotes the average error estimated only on outliers.

Param	**Attn.Q_Proj**						**Attn.K_Proj**					
Layer	**Low**		**Medium**		**High**		**Low**		**Medium**		**High**	
Type	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**
Low-Rank	0.75	2.24	4.24	14.31	4.47	10.28	0.87	9.90	4.79	34.04	4.82	31.41
BitDelta	0.97	2.48	4.66	14.48	4.84	10.01	1.09	10.34	5.16	33.03	5.14	28.06
Single	0.20	0.74	1.37	5.11	1.24	3.36	0.23	3.19	1.52	11.30	1.36	8.48
Triple	**0.13**	**0.28**	**0.54**	**1.07**	**0.71**	**0.88**	**0.15**	**0.56**	**0.58**	**1.99**	**0.73**	**2.10**
Param	**Attn.V_Proj**						**Attn.O_Proj**					
Layer	**Low**		**Medium**		**High**		**Low**		**Medium**		**High**	
Type	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**
Low-Rank	0.41	3.61	1.84	8.27	2.93	4.64	0.01	0.13	0.10	0.39	0.38	5.94
BitDelta	0.45	3.60	1.95	8.02	3.18	4.85	0.01	0.13	0.11	0.44	0.37	5.45
Single	0.14	1.42	0.65	3.58	0.79	1.45	0.00	0.04	0.03	0.10	0.10	1.60
Triple	**0.04**	**0.12**	**0.21**	**0.35**	**0.52**	**0.61**	0.00	**0.01**	**0.02**	**0.05**	**0.06**	**0.92**
Param	**FFN.Up_Proj**						**FFN.Gate_Proj**					
Layer	**Low**		**Medium**		**High**		**Low**		**Medium**		**High**	
Type	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**	**All**	**Out.**
Low-Rank	0.13	0.86	0.93	3.43	2.20	11.45	0.10	0.26	0.79	1.12	1.87	9.74
BitDelta	0.18	0.97	1.06	3.84	2.38	12.22	0.13	0.31	0.90	1.26	2.02	11.74
Single	0.03	0.17	0.27	1.08	0.56	3.10	0.02	0.06	0.23	0.35	0.47	2.14
Triple	0.03	**0.11**	**0.15**	**0.49**	**0.39**	**2.01**	0.02	**0.03**	**0.14**	**0.15**	**0.35**	**1.64**

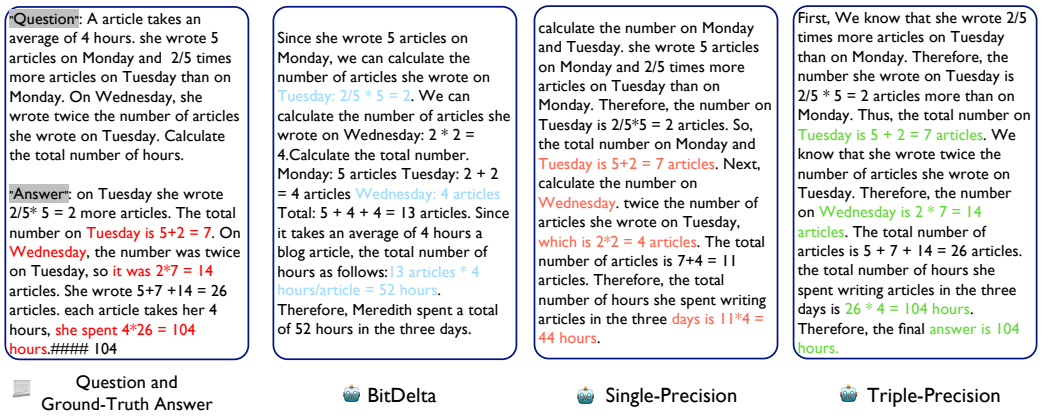


Figure 4: Case study for different delta-compression methods, where only the triple-precision compression method proposed in this work can give the correct answer.

7 Conclusion

In this paper, we propose Delta-CoMe, a delta-compression method with mixed-precision inspired by the long-tail distribution of singular values in the delta weights. Delta-CoMe achieves near-lossless performance compared to uncompressed aligned models across various typical tasks, including math, code, chat, and multi-modal tasks. We validate the effectiveness of Delta-CoMe on several widely-used aligned LLMs, whose backbone pre-trained models include Llama-2, Llama-3, and Mistral. Experimental results demonstrate that Delta-CoMe outperforms several representative baselines by a considerable margin. We believe the newly introduced Delta-CoMe method has significant value for many practical applications, such as multi-tenant serving.

https://doi.org/10.52202/079017-0978

Acknowledgments and Disclosure of Funding

This work is supported by the National Key R&D Program of China (No.2022ZD0116312), National Natural Science Foundation of China (No. 62236004, No. 62236011), and Institute Guo Qiang at Tsinghua University.

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Bablani, D., Mckinstry, J. L., Esser, S. K., Appuswamy, R., and Modha, D. S. Efficient and effective methods for mixed precision neural network quantization for faster, energy-efficient inference. *arXiv preprint arXiv:2301.13330*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dettmers, T., Svirschevski, R. A., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. In *The Twelfth International Conference on Learning Representations*, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Guo, H., Brandon, W., Cholakov, R., Ragan-Kelley, J., Xing, E. P., and Kim, Y. Fast matrix multiplications for lookup table-quantized llms, 2024. URL https://arxiv.org/abs/2407.10960.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hudson, D. A. and Manning, C. D. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

 https://doi.org/10.52202/079017-0978

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024a.

Liu, J., Xiao, G., Li, K., Lee, J. D., Han, S., Dao, T., and Cai, T. Bitdelta: Your fine-tune may only be worth one bit. *arXiv preprint arXiv:2402.10193*, 2024b.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023a.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023b.

Ryu, S., Seo, S., and Yoo, J. Efficient storage of fine-tuned models via low-rank approximation of weight residuals. *arXiv preprint arXiv:2305.18425*, 2023a.

Ryu, S., Seo, S., and Yoo, J. Efficient storage of fine-tuned models via low-rank approximation of weight residuals. *arXiv preprint arXiv:2305.18425*, 2023b.

Shen, X., Dong, P., Lu, L., Kong, Z., Li, Z., Lin, M., Wu, C., and Wang, Y. Agile-quant: Activation-guided quantization for faster inference of llms on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18944–18951, 2024.

Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., and Rohrbach, M. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8317–8326, 2019.

Tillet, P., Kung, H.-T., and Cox, D. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wang, G., Cheng, S., Zhan, X., Li, X., Song, S., and Liu, Y. Openchat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations*, 2023.

Wang, H., Ping, B., Wang, S., Han, X., Chen, Y., Liu, Z., and Sun, M. LoRA-flow: Dynamic LoRA fusion for large language models in generative tasks. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12871–12882, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.695. URL https://aclanthology.org/2024.acl-long.695.

Wei, J., Cao, S., Cao, T., Ma, L., Wang, L., Zhang, Y., and Yang, M. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge, 2024. URL https://arxiv.org/abs/2407.00088.

Wei, Y., Wang, Z., Liu, J., Ding, Y., and Zhang, L. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*, 2023.

Wu, X., Huang, S., and Wei, F. Mole: Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*, 2023.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions, 2023.

Yao, X. and Klimovic, A. Deltazip: Multi-tenant language model serving via delta compression. *arXiv preprint arXiv:2312.05215*, 2023.

Yao, Z., Dong, Z., Zheng, Z., Gholami, A., Yu, J., Tan, E., Wang, L., Huang, Q., Wang, Y., Mahoney, M., et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pp. 11875–11886. PMLR, 2021.

Yu, L., Jiang, W., Shi, H., Jincheng, Y., Liu, Z., Zhang, Y., Kwok, J., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*. PMLR, 2024.

Zhang, Z., Lei, L., Wu, L., Sun, R., Huang, Y., Long, C., Liu, X., Lei, X., Tang, J., and Huang, M. Safetybench: Evaluating the safety of large language models with multiple choice questions. *arXiv preprint arXiv:2309.07045*, 2023.

 https://doi.org/10.52202/079017-0978

A Limitation and Broader Impact

For limitations, on the one hand, we carried out extensive experiments to verify Delta-CoMe is near lossless in delta compression. However, we haven't explored mixed-precision in model compression. Recently, mixed-precision is applied widely in model compression and Delta-CoMe can provide a new perspective for model compression. On the other hand, our kernel is trivial, Wei et al. (2024) and Guo et al. (2024) have implemented more advanced kernels. We can draw on their methods to achieve higher acceleration ratios.

For broader impacts, this paper presents Delta-CoMe that mainly focuses on compression which can not only boost efficiency but also save GPU memory, may bring benefits to society.

B Genetic Search for Bits Settings

In Section 5.2, we have elaborated on the setting of different bits. All our models employ the same configuration and have demonstrated near loss-less performance, which illustrates robustness.

Allocating different numbers to different bits (e.g. 16-bit, 8-bit) is a multi-objective optimization problem. We implemented a genetic algorithm to achieve a more fine-grained search. We use the following objective function,

$$f = \min \text{PPL}(x_1, x_2, x_3, x_4, x_5)$$

where x_1, x_2, x_3, x_4, x_5 indicating the number of 16-bit, 8-bit, 4-bit, 3-bit, 2-bit and PPL$(.)$ means we calculate perplexity using samples randomly chosen form C4 dataset. Table 9 illustrates the results, particularly in code tasks, where genetic search shows a significant improvement compared to greedy search. The average performance of genetic search across all tasks even surpasses that of the original half-precision models. However, the time and storage overhead of genetic search is much greater than that of greedy search.

Table 9: The performance of different bits allocate methods on 7B aligned models. "Greedy search" represents the method in Section 5.1.

Method	α	WizardMath		MagicoderS-CL		Llama-2-chat		Llava-v1.5		**Ave.**
		GSM8K	**MATH**	**HumanEval**	**MBPP**	**SafetyBench**	**TruthfulQA**	**GQA**	**TextVQA**	
Backbone	1	11.0	2.9	38.4	47.6	41.7	38.9	n/a	n/a	n/a
Aligned	1	55.2	10.9	70.7	69.2	59.5	44.6	62.0	58.2	53.5
Greedy S.	1/16	53.6	10.3	67.1	67.9	59.8	46.9	61.7	58.5	53.2
Genetic S.	1/16	53.6	10.3	69.5	68.9	**59.9**	**47.3**	61.7	**58.5**	**53.7**

C Delta-CoMe Combine with Low-bit Backbone

Quantization methods (e.g., GPTQ, AWQ) have been widely used for quantizing backbones. It is of great significance for us to verify whether Delta-CoMe can still maintain good performance in low-bit backbone scenarios.

We evaluated the performance of Delta-CoMe using various backbones across multiple tasks in Table 10. We utilized GSM8K for math tasks, MBPP for code, TruthfulQA for chat, and TextVQA for multi-modal tasks. Table 10 has demonstrated that even when backbones are in low precision, Delta-CoMe can achieve performance similar to the original, indicating that Delta-CoMe can be further applied to backbones of various precision levels.

D Exploring the boundary of Delta-CoMe

We have shown that Delta-CoMe can maintain near lossless performance under a 16× compression ratio. In the following, we attempt to explore the compression limits of Delta-CoMe. We employ

Table 10: Performance drop in 4-bit and 16-bit backbone across different tasks.

Precision	Backbone	Tasks	Delta
4-BIT BACKBONE	WizardMath 4-bit	49.36	n/a
	Llama2 4-bit + 1bit delta	47.01	-2.3
16-BIT BACKBONE	WizardMath 16-bit	55.2	n/a
	Llama2 16-bit + 1bit delta	53.6	-1.6
4-BIT BACKBONE	Magicoder 4-bit	66.2	n/a
	Codellama-python 4-bit + 1bit delta	65.4	-0.8
16-BIT BACKBONE	Magicoder 16-bit	66.7	n/a
	Codellama-python 16-bit + 1bit delta	67.2	+0.3
4-BIT BACKBONE	WizardMath 4-bit	49.36	n/a
	Llama2 4-bit + 1bit delta	47.01	-2.3
16-BIT BACKBONE	WizardMath 16-bit	55.2	n/a
	Llama2 16-bit + 1bit delta	53.6	-1.6
4-BIT BACKBONE	Llava-v1.5 4-bit	57.68	n/a
	Vicuna 4-bit + 1bit delta	57.58	-0.1
16-BIT BACKBONE	Llava-v1.5 16-bit	58.2	n/a
	Vicuna 16-bit + 1bit delta	58.5	+0.3

WizardMath-7B in GSM8K task to carry out our experiments which is shown in 11. For all the experiments, the rank share the same setting.

When the compression ratio is within 20×, Delta-CoMe still performs well. However, at a compression ratio of 32×, there is a noticeable decline in performance, but it still outperforms low-rank and low-bit methods, which only achieve a 16× compression ratio.

Table 11: Performance under different compression ratios for WizardMath-7B

Model	w/o Comp.	1/16	1/18	1/20	1/22	1/26	1/32
WizardMath-7B	55.2	53.6	52.2	51.9	51.2	50.1	48.8

https://doi.org/10.52202/079017-0978

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we mention that singular values follow a long-tailed distribution. Based on this observation, we propose mixed-precision quantization, assigning more bits to singular vectors corresponding to larger singular values, and provide experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have elaborated on the limitations in the "Limitation and Broader Impact" section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: There are no theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provided details of the key parameters used in the experiments, set a random seed, and used greedy decoding to facilitate the reproducibility of our experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

https://doi.org/10.52202/079017-0978

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will open-source all resources once the paper is de-anonymized.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In the experiment section of our paper, we provide a setup subsection that details our experimental settings.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The proposed delta-compression method does not rely on any randomness. Repeatedly run the proposed algorithm can always yield the same compressed delta weights. We thus did not conduct statistical significance tests in our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: As reported in this paper, we use NVIDIA A100 GPUs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have thoroughly reviewed the guidelines and ensure that we follow them clearly.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Delta-CoMe can efficiently compress and deploy multiple models on a single GPU, saving the original computational cost and reducing carbon emissions.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/079017-0978

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: In this paper, our main contribution is proposing a novel compression algorithm. We have not released any potentially harmful models or data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We can ensure that we have cited correctly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: this paper does not release any new datasets or models.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: In this paper, we did not conduct crowdsourcing experiments nor involve human subjects because our research does not require them.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer:[NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

https://doi.org/10.52202/079017-0978

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.