

BitDelta: Your Fine-Tune May Only Be Worth One Bit

James Liu^{1*} Guangxuan Xiao¹ Kai Li² Jason D. Lee² Song Han^{1,3} Tri Dao^{2,4} Tianle Cai^{2,4*}

¹MIT ²Princeton University ³NVIDIA ⁴Together AI

 <https://github.com/FasterDecoding/BitDelta>

Abstract

Large Language Models (LLMs) are typically trained in two phases: pre-training on large internet-scale datasets, and fine-tuning for downstream tasks. Given the higher computational demand of pre-training, it is intuitive to assume that fine-tuning adds less new information to the model, and is thus more compressible. We explore this assumption by decomposing the weights of fine-tuned models into their pre-trained components and an additional *delta*. We introduce a simple post-fine-tuning method, BitDelta, which successfully quantizes this delta down to 1 bit without compromising performance. This interesting finding not only highlights the potential redundancy of information added during fine-tuning, but also has significant implications for the multi-tenant serving and multi-tenant storage of fine-tuned models. By enabling the use of a single high-precision base model accompanied by multiple 1-bit deltas, BitDelta dramatically reduces GPU memory requirements by more than 10 \times , thus reducing per-user generation latency by more than 10 \times in multi-tenant settings. We validate BitDelta through experiments across Llama-2, Mistral and MPT model families, and on models up to 70B parameters, showcasing minimal performance degradation in all tested settings.

1 Introduction

After large-scale pretraining, foundation models are typically fine-tuned for specific downstream tasks [16, 43, 44]. This *pretrain-finetune* paradigm has revolutionized machine learning; LLMs have not only proven effective for critical tasks such as instruction following and alignment [39], but are also performant on a wide array of niche yet highly impactful applications [61, 42]. Through fine-tuning, LLMs are adeptly equipped to align with distinct user preferences or specialized task requirements, showcasing an unprecedented level of adaptability. Thus, the prospect of serving millions of uniquely fine-tuned models, each tailored to individual tasks and user needs, presents a promising vision for the future of machine learning.

Realizing this vision is challenging due to two key reasons: 1) **Expensive Storage**. Each new fine-tuned model is large, even if we have relatively few base models, making them expensive to store and challenging to manage on disk. 2) **Expensive Serving**. Distinct fine-tuned models each demand significant GPU memory, making it difficult and expensive to concurrently serve such models without noticeable downtime. To tackle these issues, we decompose the fine-tuned model weights into the weights of the base pre-trained model and a *delta* induced by the fine-tuning process. By compressing this delta while maintaining model performance, we aim to sidestep the prohibitive costs associated with storage and GPU memory demands.

*Correspondence to jamesll@mit.edu, tianle.cai@princeton.edu. Tianle’s contribution was partially done during consulting at Together AI.

From the delta decomposition point of view, parameter-efficient fine-tuning (PEFT) methods like LoRA [25, 24, 46, 15, 9] effectively enforce a highly structured and compressed form of delta *during fine-tuning*, a powerful insight for model serving of PEFT-based fine-tunes. Sheng et al. [49] and Chen et al. [7] explore multi-tenant serving of LoRA-based fine-tunes.

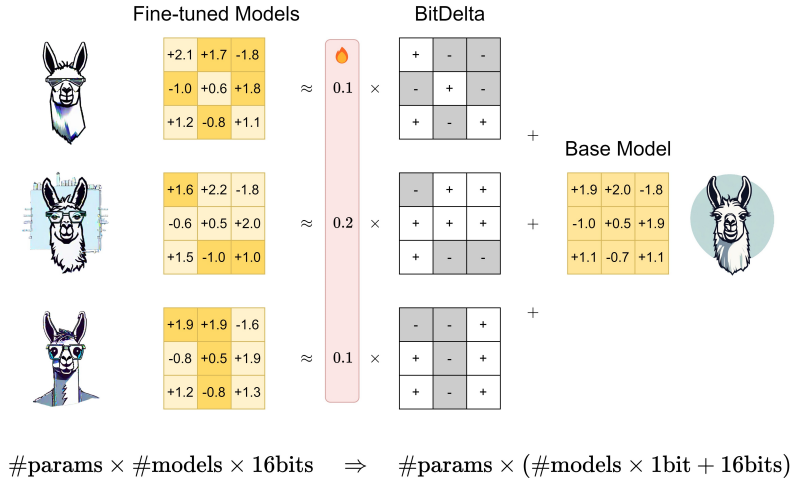


Figure 1: **Overview of BitDelta.** BitDelta applies 1-bit quantization to the weight delta between fine-tuned and base models. For each weight matrix, we quantize its delta as its sign bits and a trainable high-precision scale factor. The scale factor is initialized to achieve the best approximation error in L_2 norm and further refined with a few distillation steps. BitDelta shows minimal degradation in model performance and reduces memory consumption in multi-tenancy serving by representing multiple fine-tuned models with a single high-precision base model and multiple 1-bit deltas.

Nevertheless, recent work has shown that PEFT methods may not yet match the model quality of full parameter fine-tuning, especially on high resource tasks [6], and are fairly sensitive to hyperparameter choice and prompting methods [38]. Biderman et al. [2] show that LoRA’s reduced expressivity, although providing desirable regularization, leads to significantly worse performance compared to full fine-tuning in math and programming tasks. As a result, we notice that among the 2307 LLMs (as of time of writing) on the Open LLM Leaderboard [1] with a valid README file, only $< 20\%$ indicate that they exclusively use LoRA. Most models are full parameter fine-tunes, model merges [64, 28, 59] of full parameter fine-tunes, or model merges of LoRA based fine-tunes (which are effectively high-rank).

It is also attractive to approximate general deltas with low-rank matrices *post-training* (in particular, *post-fine-tuning*). However, experimental results show that this is challenging (Table 1), as deltas from full parameter fine-tunes tend to be fairly high-rank (Figure 2).

We instead draw from the insight that motivates PEFT methods in general: Given the higher computational demand of pre-training, it is intuitive to assume that fine-tuning adds less new information to the model, and is thus *much* more compressible. In fact, we find that we can efficiently *quantize* the delta to merely *1 bit* with almost no performance drop. We propose BitDelta, an efficient post-training quantization (PTQ) solution that acts on the weight delta between a fine-tuned model and its underlying base model.

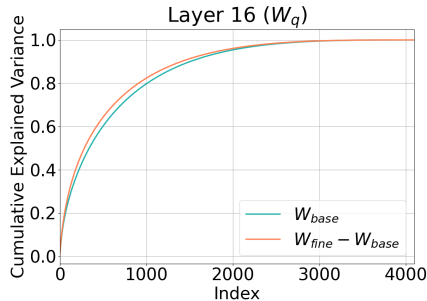


Figure 2: Cumulative Explained Variance (CEV) plot of a 4096×4096 weight delta between *Llama 2-7B* and *Vicuna-7B v1.5*. Deltas from full parameter fine-tuning are fairly high rank, making low-rank approximations difficult.

BitDelta consists of two stages: 1) We quantize the delta between a fine-tuned model’s weight matrix and base model’s weight matrix into a scaling factor multiplied by a binary matrix. Specifically, we

Table 1: Comparison between BitDelta and a SVD based method, with *Llama 2-7B* and *Llama 2-7B Chat* as the base and fine-tuned models. BitDelta is performant across the board, whereas the SVD-based method fails to sufficiently capture the fine-tuned information.

| Model/Method | TruthfulQA | GSM8K | MT-Bench | Adjusted Average [†] ↑ |
|---------------------------|------------|-------|----------|---------------------------------|
| <i>Llama 2-7B</i> | 38.96 | 13.57 | – | 60.53 |
| <i>Llama 2-7B Chat</i> | 45.32 | 22.74 | 6.56 | 59.81 |
| BitDelta-Initial | 41.10 | 18.27 | 6.31 | 60.70 |
| BitDelta | 44.95 | 20.24 | 6.47 | 59.88 |
| SVD-Initial ($r = 16$) | 42.57 | 7.13 | 4.73 | 60.58 |
| SVD ($r = 16$) | 42.42 | 5.05 | 4.99 | 60.71 |
| SVD-Initial ($r = 128$) | 43.90 | 17.82 | 5.68 | 60.21 |
| SVD ($r = 128$) | 43.32 | 11.83 | 5.85 | 60.58 |

take the sign of the weight delta to form the binary matrix and initialize the scaling factor as the average of the absolute values of the delta, minimizing L_2 quantization error. 2) We further calibrate the scaling factors through model distillation over a small calibration dataset while keeping the binary matrices frozen. Despite the small number of trainable parameters and calibration steps, we find that this distillation process is effective in further recovering model quality. Our experiments over 17 popular fine-tuned models affirm that BitDelta can be applied across various model types and model sizes with minimal impact on performance.

BitDelta creates opportunities to efficiently serve multiple fine-tuned models with shared servers: By only storing a single full-precision base model, and (dynamically) loading and performing batched inference over multiple 1-bit deltas, we can efficiently represent multiple fine-tuned models. Compared to naively using full precision fine-tuned models, deltas compressed by BitDelta are more than $10\times$ smaller, and can therefore be loaded faster. This addresses the storage challenge. Moreover, since LLM inference is memory-bound [32, 5, 3], the latency of each decoding step is proportional to the GPU memory consumption of the model weights. With an efficient CUDA kernel implementation, we can translate this memory reduction into a latency reduction, similar to other quantization methods [19, 33]. Using the $W_{INT1}A_{FP16}$ kernel from BitBLAS [58], we improve the multi-tenant serving latency of full-parameter fine-tuned models by more than $10\times$.

Finally, we study a few extensions of BitDelta, where we quantize the base model and where we iteratively apply BitDelta. Experimental results show that our method is quite general and can be applied to various use cases.

2 Related Work

2.1 Full Model Compression

Quantization. Quantization techniques are widely used to reduce memory consumption and improve LLMs’ generation latency. Xiao et al. [60] implement a technique that rescales between activations and parameters, effectively mitigating outlier activations to facilitate smoother quantization. Dettmers et al. [14] develop an approach that decomposes matrix multiplications into 8-bit computations, with an additional 16-bit process for handling outliers. Exploring further, Frantar et al. [19] introduce a method that iteratively rounds weight columns to 3-4 bits of precision. Similarly, Lin et al. [33] propose an activation-aware quantization scheme that selectively preserves crucial weights while compressing the majority to 3-4 bits. Kim et al. [29] devise a sparse, low-precision pattern focusing on a small yet significant set of weights. Chee et al. [4] utilize incoherence processing to quantize model weights to as low as 2 bits with minimal impact on performance.

Pruning. Pruning also aims to reduce the memory consumption of neural networks. It accomplishes this by pushing certain parameter values to zero, inducing sparsity in the model [31, 21, 22, 67]. However, these methods may fail to take advantage of modern hardware like GPUs unless using

[†]Adjusted Average is over ARC, BBH, HellaSwag, WinoGrande, and excludes TruthfulQA, GSM8K, MT-Bench.

certain structured sparsity patterns like 2:4 (50%) sparsity [36]. Frantar and Alistarh [18] demonstrate a pruning method on LLMs that successfully utilizes the 2:4 sparsity pattern and achieves a 50% sparsity ratio. It is challenging to obtain higher sparsity while being hardware-friendly.

Early work on post-training delta compression. Most related to our work, a few studies explore the idea of post-training delta compression by adopting existing compression techniques like GPTQ, unstructured pruning [22], or even classic lossless compression algorithms. Isik et al. [26] focus on reducing the delta size to save storage. Yu et al. [64] utilize pruning to improve model merging applications. Yadav et al. [62] reduces the size of PEFT modules to save storage. Ryu et al. [47] combines quantization with a low-rank approximation to reduce the delta size. The concurrent and independent work by Yao and Klimovic [63] also explores using delta compression to improve multi-tenant serving, but focuses more on reducing the model loading time from disk to GPU. Compared to existing work, we offer a much simpler and faster method, BitDelta, achieving a compression ratio of more than $10\times$ while also being friendly to modern accelerators.

3 BitDelta

3.1 Method

BitDelta consists of two stages: 1) We quantize each weight matrix into a scalar multiplied by a binary matrix[†]. 2) We further calibrate the scalar factors using model distillation. We describe each stage in this section:

1-bit quantization. Let $W_{\text{base}}, W_{\text{fine}} \in \mathbb{R}^{n \times m}$ be weight matrices from the base model and fine-tuned model respectively. We define the weight delta as $\Delta = W_{\text{fine}} - W_{\text{base}}$, representing the modification in weights post-fine-tuning. For efficient representation of this weight delta, we aim to obtain a binarized estimator by encoding its sign bits, denoted as $\hat{\Delta}$:

$$\hat{\Delta} = \alpha \odot \text{Sign}(\Delta), \tag{1}$$

where

$$\text{Sign}(W_{ij}) = \begin{cases} +1, & \text{if } W_{ij} > 0, \\ -1, & \text{if } W_{ij} \leq 0, \end{cases} \tag{2}$$

and α is a high-precision scaling factor for the entire matrix. To minimize the quantization error of Δ in L_2 norm:

$$\|\Delta - \hat{\Delta}\|_2^2 = \sum_{ij} (|W_{ij}| - \alpha)^2, \tag{3}$$

we initialize α as follows:

$$\alpha = \frac{1}{nm} \sum_{ij} |\Delta_{ij}|. \tag{4}$$

Surprisingly, we find that the above quantization approach already does quite well and retains most of the fine-tuned models’ performance.

Scale distillation. The scaling factor α intuitively plays a more significant role in the low-bit regime. Additionally, per-matrix L_2 weight error is not a perfect measure of degradation in *overall* model quality. We further optimize these scales by performing model distillation to align the output logits of the quantized model to that of the original fine-tuned model. More concretely, we freeze the model weights and optimize for the following objective:

$$\alpha^* = \arg \min_{\alpha} \mathbb{E}_{x \sim \mathbf{X}} \left[\|\mathbf{Z}_{\text{fine}}(x) - \mathbf{Z}_{\text{bin}}(x; \alpha)\|^2 \right] \tag{5}$$

[†]In our experiments, we only quantize the linear layers in the Transformer blocks as they contribute the majority of the parameters and computation.

Table 2: BitDelta works on Llama-2 and Mistral families and on a wide range of model sizes ranging from 7B to 70B parameters. BitDelta works for many types of fine-tuned information, including SFT-based methods, RLHF-based methods, and context extension methods (RoPE scaling). Scale distillation is effective, raising TruthfulQA/GSM8K scores to within 1-2 points of the baseline fine-tune, and MT-Bench scores to within 0.1-0.2 points.

| Model | Method | TruthfulQA | GSM8K | MT-Bench | Adjusted Average [†] \uparrow |
|----------------------------|------------------|------------|-------|----------|--|
| <i>Llama 2-7B</i> | – | 38.96 | 13.57 | – | 60.53 |
| <i>Llama 2-7B Chat</i> | Baseline | 45.32 | 22.74 | 6.56 | 59.81 |
| | BitDelta-Initial | 41.10 | 18.27 | 6.31 | 60.7 |
| | BitDelta | 44.95 | 20.24 | 6.47 | 59.88 |
| <i>Vicuna-7B v1.5 16k</i> | Baseline | 50.38 | 14.18 | 6.06 | 57.50 |
| | BitDelta-Initial | 45.58 | 13.95 | 5.69 | 58.51 |
| | BitDelta | 48.75 | 14.48 | 6.24 | 57.64 |
| <i>Llama 2-13B</i> | – | 36.90 | 22.74 | – | 64.68 |
| <i>Llama 2-13B Chat</i> | Baseline | 43.95 | 33.13 | 6.98 | 63.99 |
| | BitDelta-Initial | 41.70 | 33.36 | 7.06 | 64.25 |
| | BitDelta | 43.47 | 31.92 | 6.95 | 63.96 |
| <i>Vicuna-13B v1.5 16k</i> | Baseline | 50.38 | 29.72 | 6.90 | 57.5 |
| | BitDelta-Initial | 41.7 | 26.76 | 6.60 | 64.25 |
| | BitDelta | 48.75 | 28.73 | 6.88 | 57.64 |
| <i>WizardLM-13B v1.2</i> | Baseline | 47.17 | 42.38 | 6.95 | 61.61 |
| | BitDelta-Initial | 44.89 | 42.08 | 6.73 | 61.91 |
| | BitDelta | 46.67 | 41.62 | 6.93 | 61.86 |

where \mathbf{X} is a calibration dataset, and $\mathbf{Z}(\cdot)$ are the logits of the respective models. Scale distillation is fairly robust to choice \mathbf{X} , as 1) the process is extremely parameter efficient, and 2) the crucial aspect of the process is to logit match with the fine-tuned model, regardless of the actual text content.

For our experiments, we distill on the C4 dataset [45], consisting of generic internet data, using 800 samples of length 128. We use the same subset of C4 over all models to control for seed-based variations. We use the Adam optimizer [30] with $lr = 10^{-4}$, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$. 1x80 GB A100 GPU is used to distill 7B and 13B models, and 6x80GB A100 GPUs are used to distill 70B models (2x for finetune, 4x for binarized). Scale distillation is fast; we can compress 70B models in roughly 10 minutes.

3.2 Methodology Cost

Compared to full parameter and parameter efficient fine-tuning methods, BitDelta is extremely cheap. While fine-tuning methods require training thousands to millions of parameters, BitDelta only necessitates training a single parameter per weight matrix. Moreover, BitDelta operates efficiently with input sequences of length 128, unlike fine-tuning methods that demand longer sequences to saturate the context window (4k, 8k, etc.). Crucially, BitDelta requires only 200 training steps (assuming a batch size of 4), which is significantly less compared to the 10000-1000000 steps at higher batch sizes needed by fine-tuning methods. Thus, in terms of methodology cost, we liken BitDelta more to post-training quantization (PTQ) schemes like GPTQ [19] and AWQ [33], rather than full parameter or parameter efficient fine-tuning, while being faster than most PTQ schemes.

3.3 Implication

The ability to compress the delta to merely 1-bit opens up multiple opportunities for improving efficiency, enabling more effective model storage [26] – where a single base model can be maintained alongside multiple compressed deltas – and facilitating model hot-swapping [7, 49]. With hot-swapping, the base model remains in GPU memory, and compressed deltas are dynamically loaded in accordance to incoming requests. In both cases, the compression ratio can be directly translated into reductions in storage needs and loading times.

Moreover, BitDelta enables the possibility of a multi-tenant serving system like Punica [7] or S-LoRA [49] but for general fine-tuned models instead of just LoRA models. Concretely, we consider the scenario where multiple models fine-tuned from the same base model are served with the same

Table 3: Continuation of Table 2.

| Model | Method | TruthfulQA | GSM8K | MT-Bench | Adjusted Average† ↑ |
|---------------------------------|------------------|------------|-------|----------|---------------------|
| <i>Llama 2-70B</i> | – | 44.82 | 52.69 | – | 71.81 |
| <i>Llama 2-70B Chat</i> | Baseline | 52.77 | 47.61 | 7.12 | 68.82 |
| | BitDelta-Initial | 41.63 | 42.38 | 6.85 | 66.01 |
| | BitDelta | 51.37 | 48.82 | 7.06 | 69.14 |
| <i>Solar-0-70B</i> | Baseline | 62.03 | 56.18 | 7.07 | 73.77 |
| | BitDelta-Initial | 59.08 | 56.79 | 6.79 | 73.14 |
| | BitDelta | 62.03 | 56.63 | 6.82 | 73.57 |
| <i>Mistral-7B v0.1</i> | – | 42.60 | 37.76 | – | 65.98 |
| <i>Mistral-7B v0.1 Instruct</i> | Baseline | 55.93 | 32.75 | 6.86 | 60.36 |
| | BitDelta-Initial | 51.27 | 38.82 | 6.54 | 63.83 |
| | BitDelta | 55.23 | 31.54 | 6.43 | 61.10 |
| <i>Zephyr-7B-β</i> | Baseline | 55.12 | 34.34 | 7.18 | 65.22 |
| | BitDelta-Initial | 54.53 | 40.26 | 6.70 | 66.12 |
| | BitDelta | 58.39 | 31.92 | 7.00 | 66.20 |
| <i>Dolphin 2.2.1</i> | Baseline | 54.02 | 54.28 | 7.36 | 67.31 |
| | BitDelta-Initial | 48.14 | 50.27 | 7.10 | 67.58 |
| | BitDelta | 54.91 | 52.84 | 7.20 | 66.97 |
| <i>MPT-7B</i> | – | 33.37 | 6.22 | – | 57.95 |
| <i>MPT 7B-Chat</i> | Baseline | 40.22 | 7.96 | 5.00 | 56.5 |
| | BitDelta-Initial | 38.96 | 10.01 | 4.39 | 57.11 |
| | BitDelta | 39.87 | 8.11 | 4.94 | 56.52 |

server. This setting greatly exploits the GPU resource and saves each fine-tuned model’s inference cost when their traffic is low or unbalanced. With BitDelta, we can keep one high-precision base model with multiple compressed deltas in the GPU memory. Compared to directly serving multiple fine-tuned models, this approach greatly saves memory consumption.

Since LLM inference follows the memory-bound computation pattern where the generation latency is proportional to the GPU memory used by the model weights, the lower memory consumption also suggests the opportunity to improve the serving latency. For example, Punica and S-LoRA exploit LoRA’s structure and memory saving by computing the activation product between the shared base weight, and low-rank fine-tuned delta weights separately. Similarly, we decompose the forward pass of each linear layer as follows:

$$X'_i = W_{\text{fine},i} X_i \approx W_{\text{base}} X_i + \underbrace{\hat{\Delta}_i X_i}_{\text{Kernel}} \quad (6)$$

where X_i and X'_i represent input and output features to the i -th fine-tuned model, and the base model weight and the 1-bit delta are computed separately. For a batch of requests, $W_{\text{base}} X_i$ can be computed with the classic batched GEMM kernel. We utilize the BitBLAS [58] $W_{INT1} A_{FP16}$ kernel that allows us to calculate $\hat{\Delta}_i X$ in a batched setting while keeping the 1-bit deltas quantized until they are transferred to the GPU cache. This kernel fuses the dequantization operation with the GEMM calculation, reducing the data moving overhead by a large factor.

4 Experiments

4.1 Setup

Baselines. Our primary baselines are the original fine-tuned models without compression. We also compare with 8-bit RTN, 4-bit GPTQ [19], and 2-bit QuIP# [54] on evaluations where we run BitDelta on quantized base models.

Models and datasets. We benchmark fine-tuned models based on the Llama-2 [53], Mistral [27], and MPT [51] model families: Vicuna, Xwin-LM, Solar-70B, Zephyr, OpenChat 3.5, Dolphin 2.2.1, and OpenOrca [10, 52, 56, 55, 57, 23, 37]. We evaluate on eight tasks: MT-Bench, 25-shot ARC Challenge, 5-shot BBH, 10-shot HellaSwag, zero-shot TruthfulQA, zero-shot LAMBADA, zero-shot

Table 4: Comparison of model responses from *Zephyr-7B-β* for Question 9 in MT-Bench, a concise advertisement task. BitDelta-Initial is unable to follow the instructions, producing an advertisement that is overly formal and makes no attempt to adhere to the word limit. With the addition of scale distillation, BitDelta successfully produces a concise, catchy advertisement slightly over the word limit. *Prompt slightly modified for clarity.

| | |
|-------------------------|---|
| Prompt | Please create a slogan about bioenergy that satisfies the following: 1. Make the tone sound casual 2. Embed an advertisement for a company called "FlexPower" 3. Fewer than 10 words. |
| BitDelta | "FlexPower's groundbreaking bioenergy solution: Produce fuel without compromising food production!" (10 words) This headline is catchy, casual, and highlights the unique selling point of the... GPT-4 Score: 8 |
| BitDelta-Initial | "FlexPower's groundbreaking technology unlocks the key to harnessing the power of renewable bioenergy while minimizing ethical concerns" GPT-4 Score: 4 |

Winogrande, and 5-shot GSM8K [66, 12, 50, 65, 34, 40, 48, 13]. We use FastChat [66] to evaluate on MT-Bench, and use `lm-evaluation-harness` [20] to evaluate on the other tasks. We denote our methodology before scale distillation is applied as BitDelta-Initial.

We primarily focus on high-margin metrics where fine-tuning is significantly impactful and aggregate the other metrics. See Tables 7 to 10 in the Appendix for full results. BitDelta performs quite well on the aggregated metrics, even outperforming the baseline in many cases. However, it's important to contextualize these results with regard to the base model itself, which is also performant on these metrics. It's difficult to attribute performance to our methodology or to the underlying base model in such cases. Because of this, we highlight TruthfulQA, GSM8K, and MT-Bench, which base models tend to struggle on, to show that BitDelta accurately preserves fine-tune information.

4.2 Accurate Quantization

SVD comparison. We compare BitDelta to a low rank approx. of the weight delta on *Vicuna-7B v1.5*. For the low rank approx., we decompose $\Delta = U\Sigma V$ and approximate $\hat{\Delta} = AB$ where $A = U\sqrt{\hat{\Sigma}}$, $B = \sqrt{\hat{\Sigma}}V$. During distillation, we treat all entries of the low rank matrices as trainable parameters. We compare against two settings: $r = 16$ (most commonly used) and $r = 128$ (memory equivalence with BitDelta). We find that the low rank approx. fails to fully capture the fine tune information, and underperforms across the board (Table 1). In particular, the low rank approx. heavily underperforms on MT-Bench [10], a difficult multi-turn instruction following dataset fairly indicative of real world performance. Interestingly, distillation is not as effective for the low rank approx. compared to BitDelta.

Main Results. BitDelta is performant across various model families, across a wide range of model sizes, and across many fine-tuning techniques. We benchmark on Llama-2, Mistral, and MPT, families, and on models ranging from 7B to 70B parameters. Shown in Table 2, we find that BitDelta is very general and can recover all types of finetune information, including SFT-based methods [43] on *Mistral-7B v0.1 Instruct*, RLHF-based methods [11] on *Llama 2 Chat*, and context extension methods (RoPE scaling) [8, 41] on *Vicuna-7B v1.5 16k*.

We note that GSM8K for BitDelta-Initial on *Mistral-7B v0.1 Instruct* and *Zephyr-7B-β* is abnormally high; we attribute this to how performant the base model *Mistral-7B v0.1* is on this task in comparison. Scale distillation is effective, raising TruthfulQA and GSM8K scores to within 1-2 points of the baseline fine-tune, and generally raising MT-Bench scores to within 0.1-0.2 points.

Table 5: BitDelta achieves over 10× compression. We can further compress the embedding and LM head layers, but leave this to future work due to inconsistencies in tokenizer vocabularies.

| Base Model | Size | Δ Size | Comp. Factor |
|------------------------|-----------|---------------|--------------|
| <i>Llama 2-7B</i> | 13.48 GB | 1.24 GB | 10.87 |
| <i>Llama 2-13B</i> | 26.03 GB | 2.09 GB | 12.45 |
| <i>Llama 2-70B</i> | 137.95 GB | 8.95 GB | 15.41 |
| <i>Mistral-7B v0.1</i> | 14.48 GB | 1.30 GB | 11.14 |

Table 6: We apply BitDelta to *Llama 2-7B Chat* (with corresponding base model *Llama 2-7B*), and find it holds up when the underlying base model is quantized at various levels.

| Base Model | Method | TruthfulQA | GSM8K | MT-Bench | Adjusted Average [†] \uparrow |
|-------------------|---------------------|------------|-------|----------|--|
| Baseline | FP16 | 45.32 | 22.74 | 6.56 | 59.81 |
| | INT8 RTN | 45.02 | 22.29 | 6.28 | 59.63 |
| | GPTQ | 44.92 | 19.48 | 5.90 | 58.67 |
| | QuIP# | 43.69 | 10.77 | 5.37 | 55.82 |
| <i>Llama 2-7B</i> | FP16 + Δ | 44.95 | 20.24 | 6.47 | 59.88 |
| | INT8 RTN + Δ | 44.71 | 19.86 | 6.16 | 59.85 |
| | GPTQ + Δ | 42.52 | 19.94 | 6.02 | 59.22 |
| | QuIP# + Δ | 42.00 | 9.72 | 4.96 | 57.44 |

Case Study. We present a sample response from *Zephyr-7B- β* in Table 4, highlighting the efficacy of scale distillation. BitDelta-Initial does not have a casual tone, and makes no attempt to adhere to the word limit. With the introduction of scale distillation, BitDelta exhibits greater instruction following capabilities, producing a catchy response that slightly exceeds the word limit.

Quantized base models. Because 8-bit RTN, GPTQ, and QuIP# work with 16-bit activations, we can keep the fine-tune weights W_{fine} and scaling factors α in high precision in the compression process, only quantizing the base weights W_{base} . As shown in Table 6, we find that BitDelta is still performant when applied to quantized base models.

Ablation over fidelity of Δ . By successively applying BitDelta, treating the compressed model from the previous iteration as our base model, we can vary the granularity over the delta, associating it with multiple 1-bit masks. One advantage of doing this is the ability to assign arbitrary scale factors to each 1-bit mask. In contrast, when increasing the bit size, scale factors are implicitly fixed with respect to each other. Figure 3 shows how the TruthfulQA of *Llama 2-7B* plus an increasingly granular delta approaches that of *Vicuna-7B v1.5*. Full results are in Table 9.

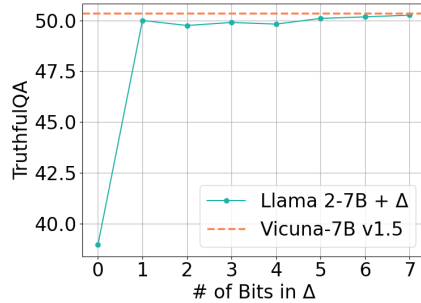


Figure 3: As the fidelity of Δ increases, the TruthfulQA scores of *Llama 2-7B* + Δ approaches that of *Vicuna-7B v1.5*.

4.3 Latency Improvement

For simplicity, we consider the setting where each model receives one distinct request simultaneously. It would be insightful to develop more sophisticated serving systems, which we leave to future work. Following the decomposition in Eq. (6), the $W_{INT1AFP16}$ kernel is used to compute the batched matrix multiplication between B binary matrices ($N \times M$) and B high-precision activations ($L \times N$) where N, M are intermediate dimensions and L is the sequence length. We focus on decoding latency which dominates runtime, as opposed to prefill latency. Tokens are generated one by one when decoding, meaning L is always 1. For all latency experiments we use a single A100 80GB with power limit set to 500W.

Kernel latency. We benchmark the decoding latency of our kernel, a batched linear operation over multiple 1-bit deltas, corresponding to the delta component of Eq. (6). We compare this to the S-LoRA kernel, a batched linear operation over multiple low-rank deltas, and also compare this to the base weight backbone shared over all deltas. We set $r = 128$ for S-LoRA, to maintain memory equivalence with BitDelta at $N = M = 4096$.

We profile the latency of the backbone ($W_{\text{base}}X$) and deltas (ΔX) separately. Although X 's memory footprint scales with batch size, it is negligible compared to W_{base} , which remains constant. For typical low to medium batch settings, which is typical for $B \times N \ll N \times M$. In such settings, the overall memory footprint of the backbone is effectively independent of batch size, as shown in Figure 4 (left). This is in contrast with that of the deltas, which scales with the batch size, as each

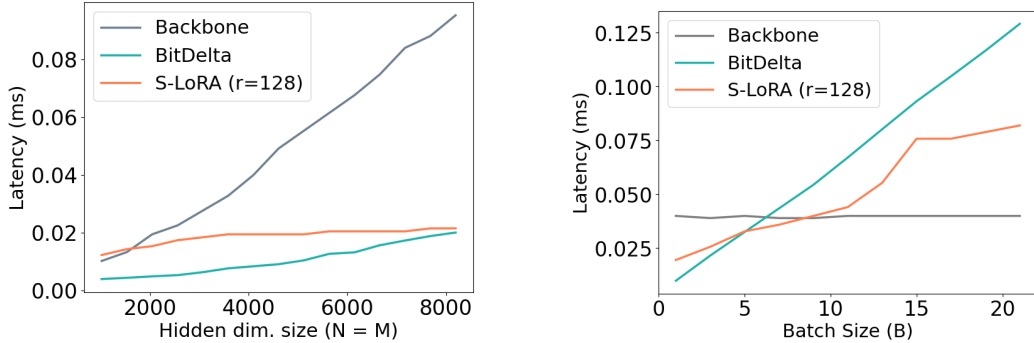


Figure 4: Decoding latency of a linear layer, as in Eqn. 6. Black: Shared base weight backbone $W_{\text{base}}X$. Blue: Batched activation-product with B 1-bit deltas, as in BitDelta. Red: Batched activation-product with B low-rank deltas, as in S-LoRA. Left: Ablation over hidden size, assuming $N = M$ and $B = 1$. Right: Ablation over batch size, assuming $N = M = 4096$.

additional client in the batch adds an additional delta. At batch size 1 (Figure 4, right), backbone latency dominates over delta latency (BitDelta and S-LoRA) due to W_{base} 's $16\times$ larger memory footprint compared to a single delta. As the batch size increases (Figure 4, left), the combined memory footprint of multiple deltas exceeds W_{base} around $B = 6$ to $B = 8$.

BitDelta underperforms slightly compared to S-LoRA in large-batch settings as the LoRA kernel is highly optimized for GPU. We emphasize that closing or even surpassing the gap is tractable. For example, Ma et al. [35] point out that $W_{INT1}A_{FP16}$ requires no multiplication operations and that new hardware can be co-designed with this in mind to drastically reduce energy/latency costs.

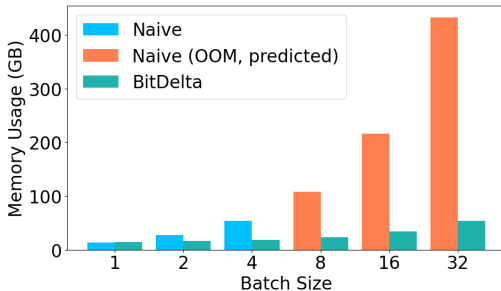


Figure 5: Memory usage of *Llama 2-7B*, assuming each sequence in the batch has a length of 128. Blue: Memory usage of the naive method, separately storing B distinct fine-tuned models. Orange: Projected values for the naive method. Green: Memory usage of BitDelta. The naive forward pass succumbs to GPU memory issues at higher batch sizes.

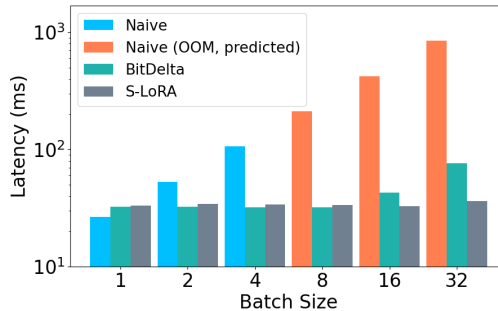


Figure 6: End-to-end decoding latency of *Llama 2-7B*. Blue: Naive forward pass with B distinct fine-tuned models. Orange: Projected values for the naive forward pass. Green: Batched forward pass with BitDelta. Gray: Batched forward pass with S-LoRA. The naive forward pass succumbs to GPU memory issues at higher batch sizes.

End-to-end latency. We benchmark the end-to-end decoding latency on *Llama 2-7B* variants with an input length of 128 (we find the decoding latency is less sensitive to the input length), ablated across batch size. For BitDelta and S-LoRA, the forward pass consists of the addition of two components: a single backbone pass (batch independent) and a delta pass (scales with batch size).

We compare BitDelta and S-LoRA with a naive method that computes each $W_i X_i$ separately in the forward pass. This naive approach scales poorly with batch size as it effectively maintains a separate backbone (W_i) for each client in the batch. Given the substantial memory footprint of the backbone, this leads to significant memory usage as batch size increases. In contrast, BitDelta and S-LoRA share a single backbone across all clients in the batch, with only the $16\times$ smaller deltas scaling with batch size. This allows for more efficient memory utilization and better performance at larger batch sizes.

We find that BitDelta and S-LoRA introduce overhead when the batch size is low. However, BitDelta and S-LoRA scale better and successfully translate the saved GPU memory to improved decoding latency, starting at $B = 2$. This is exacerbated at larger batch sizes, where the naive approach succumbs to out-of-memory issues and BitDelta and S-LoRA are still performant. In the $B \geq 16$ regime, used in modern serving solutions, BitDelta has a $>10\times$ lower per-user decoding latency than the naive method.

5 Conclusion

We propose BitDelta, a simple but effective approach to efficiently quantifying the weight delta arising from the fine-tuning of LLMs down to 1 bit. BitDelta encodes the sign bits of the weight delta and a per-weight matrix scaling factor, which is calibrated further through distillation. This allows for representing multiple full-parameter fine-tuned models with one base model and multiple 1-bit deltas, enhancing applications in multi-tenancy serving by reducing GPU memory requirements and improving generation latency. BitDelta is fast and accurate, showcasing minimal performance degradation, and opens new avenues for efficient model deployment and resource utilization in machine learning.

Acknowledgments and Disclosure of Funding

We thank Together AI, MyShell AI, National Science Foundation (NSF), MIT-IBM Watson AI Lab, MIT AI Hardware Program, and MIT Amazon Science Hub for supporting this research. JDL acknowledges support of NSF CCF 2002272, NSF IIS 2107304, NSF CIF 2212262, ONR Young Investigator Award, and NSF CAREER Award 214494. KL acknowledges the support from Meta, DataX grant from Princeton University’s Center for Statistics and Machine Learning, and innovation grant from Princeton’s School of Engineering and Applied Science.

References

- [1] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.
- [2] Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less, 2024.
- [3] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv: 2401.10774*, 2024.
- [4] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*, 2023.
- [5] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. February 2023. doi: 10.48550/ARXIV.2302.01318.
- [6] Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. Revisiting parameter-efficient tuning: Are we really there yet?, 2022.
- [7] Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant lora serving, 2023.
- [8] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.
- [9] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models, 2023.

- [10] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- [11] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.
- [12] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.
- [13] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [14] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.
- [15] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [17] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.
- [18] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023.
- [19] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.
- [20] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL <https://zenodo.org/records/10256836>.
- [21] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.
- [22] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.
- [23] Eric Hartford. Cognitivecomputations/dolphin-2.2.1-mistral-7b, hugging face, 2023. URL <https://huggingface.co/cognitivecomputations/dolphin-2.2.1-mistral-7b>.
- [24] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.
- [25] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ICLR*, 2021.
- [26] Berivan Isik, Hermann Kumbong, Wanyi Ning, Xiaozhe Yao, Sanmi Koyejo, and Ce Zhang. GPT-zip: Deep compression of finetuned large language models. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL <https://openreview.net/forum?id=h00c2tG2xL>.

- [27] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth e Lacroix, and William El Sayed. Mistral 7b, 2023.
- [28] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models, 2023.
- [29] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.
- [30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [31] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.
- [32] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. November 2022. doi: 10.48550/ARXIV.2211.17192.
- [33] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.
- [34] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.
- [35] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024.
- [36] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv: 2104.08378*, 2021.
- [37] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.
- [38] Artur Niederfahrenheit, Kouros Hakhmaneshi, and Rehaan Ahmad. Fine-tuning llms: In-depth analysis with llama-2, Sep 2023. URL <https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2>.
- [39] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- [40] Denis Paperno, Germ n Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fern andez. The lambada dataset: Word prediction requiring a broad discourse context, 2016.
- [41] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.
- [42] Jianing Qiu, Lin Li, Jiankai Sun, Jiachuan Peng, Peilun Shi, Ruiyang Zhang, Yinzhao Dong, Kyle Lam, Frank P.-W. Lo, Bo Xiao, Wu Yuan, Ningli Wang, Dong Xu, and Benny Lo. Large ai models in health informatics: Applications, challenges, and the future. *IEEE Journal of Biomedical and Health Informatics*, 27(12):6074–6087, 2023. doi: 10.1109/JBHI.2023.3316750.
- [43] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

- [44] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [45] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.
- [46] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.
- [47] Simo Ryu, Seunghyun Seo, and Jaejun Yoo. Efficient storage of fine-tuned models via low-rank approximation of weight residuals, 2023.
- [48] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.
- [49] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*, 2023.
- [50] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- [51] MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms., 2023. URL <https://www.databricks.com/blog/mpt-7b>.
- [52] Xwin-LM Team. Xwin-lm, 9 2023. URL <https://github.com/Xwin-LM/Xwin-LM>.
- [53] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [54] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks, 2024.
- [55] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of lm alignment, 2023.
- [56] Upstage. Upstage/solar-0-70b-16bit · hugging face, 2023. URL <https://huggingface.co/upstage/SOLAR-0-70b-16bit>.
- [57] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data, 2023.
- [58] Lei Wang, Lingxiao Ma, Shijie Cao, Quanlu Zhang, Jilong Xue, Yining Shi, Ningxin Zheng, Ziming Miao, Fan Yang, Ting Cao, Yuqing Yang, and Mao Yang. Ladder: Enabling efficient low-precision deep learning computing through hardware-aware tensor transformation. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 307–323, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL <https://www.usenix.org/conference/osdi24/presentation/wang-lei>.
- [59] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022.
- [60] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

- [61] Minrui Xu, Hongyang Du, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shiwen Mao, Zhu Han, Abbas Jamalipour, Dong In Kim, Xuemin Shen, Victor C. M. Leung, and H. Vincent Poor. Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aige services. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2024. doi: 10.1109/COMST.2024.3353265.
- [62] Prateek Yadav, Leshem Choshen, Colin Raffel, and Mohit Bansal. Compeft: Compression for communicating parameter efficient updates via sparsification and quantization, 2023.
- [63] Xiaozhe Yao and Ana Klimovic. Deltazip: Multi-tenant language model serving via delta compression. *arXiv preprint arXiv:2312.05215*, 2023.
- [64] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2023.
- [65] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.
- [66] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.
- [67] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *International Conference on Learning Representations*, 2017.

A Appendix

A.1 Societal Impact

Democratization of Fine-tuned Models. By dramatically reducing the hardware requirements for serving fine-tuned models, BitDelta enables smaller entities to deploy state-of-the-art models more feasibly. This can accelerate innovation and application development across various industries and academic fields, making fine-tuned models accessible to a wider audience.

Dealignment Mitigation. BitDelta is a lossy compression method on the fine-tune information in LLMs. As such, crucial alignment information may be lost in the process of compression. We believe this is an important consequence to highlight, as BitDelta democratizes multi-tenant applications which may exacerbate this dealignment concern. We encourage further work on evaluation techniques to detect alignment loss in BitDelta, which can lead to the creation of robust methods for its mitigation.

A.2 Additional Experiments

Table 7: We train a $r = 16$ LoRA finetune of *Llama 2-7B* on 1 epoch of UltraChat [17] and apply BitDelta with minimal performance degradation. This further shows the generality of BitDelta, which works on parameter-efficient fine-tunes in addition to full-parameter fine-tunes.

| Model/Method | ARC | BBH | HellaSwag | TruthfulQA | LAMBADA | WinoGrande | GSM8K | Average \uparrow | MT-Bench |
|-----------------------------|-------|-------|-----------|------------|---------|------------|-------|--------------------|----------|
| <i>Llama 2-7B</i> | 52.56 | 33.76 | 78.96 | 38.96 | 68.39 | 68.98 | 13.57 | 50.74 | – |
| <i>Llama 2-7B UltraChat</i> | 54.52 | 34.14 | 78.99 | 46.84 | 70.83 | 69.53 | 14.71 | 52.79 | 4.93 |
| BitDelta | 54.61 | 34.28 | 79.10 | 46.60 | 70.58 | 69.30 | 15.16 | 52.80 | 4.87 |

Table 8: Full results of the application of BitDelta to quantized base models, corresponding to Table 6.

| Base Model | Method | ARC | BBH | HellaSwag | TruthfulQA | LAMBADA | WinoGrande | GSM8K | Average \uparrow | MT-Bench |
|------------------------|-----------------------|-------|-------|-----------|------------|---------|------------|-------|--------------------|----------|
| Baseline | FPI16 | 53.58 | 33.84 | 78.58 | 45.32 | 66.58 | 66.46 | 22.74 | 52.44 | 6.56 |
| | LLM.int8() | 53.24 | 33.71 | 78.62 | 45.02 | 66.5 | 66.06 | 22.29 | 52.21 | 6.28 |
| | GPTQ | 51.88 | 33.54 | 77.17 | 44.92 | 65.32 | 65.43 | 19.48 | 51.11 | 5.90 |
| <i>Llama 2-7B</i> | FPI16 + Δ | 54.44 | 33.85 | 78.31 | 44.95 | 66.66 | 66.14 | 20.24 | 52.08 | 6.47 |
| | LLM.int8() + Δ | 53.67 | 33.48 | 78.57 | 44.71 | 66.7 | 66.85 | 19.86 | 51.98 | 6.16 |
| | GPTQ + Δ | 51.45 | 33.90 | 78.06 | 42.52 | 66.85 | 65.82 | 19.94 | 51.22 | 6.02 |
| <i>Llama 2-7B Chat</i> | GPTQ + Δ | 52.56 | 33.65 | 77.54 | 44.63 | 65.81 | 66.30 | 22.14 | 51.80 | 6.11 |

Table 9: Full results of the ablation over the fidelity of Δ , corresponding to Figure 3.

| # bits in Δ | ARC | BBH | HellaSwag | TruthfulQA | LAMBADA | WinoGrande | GSM8K | Average \uparrow |
|-----------------------|-------|-------|-----------|------------|---------|------------|-------|--------------------|
| <i>Llama 2-7b</i> | 52.56 | 33.76 | 78.96 | 38.96 | 68.39 | 68.98 | 13.57 | 50.74 |
| 1 bit | 54.27 | 36.57 | 77.90 | 49.97 | 65.20 | 69.46 | 20.17 | 53.36 |
| 2 bits | 54.44 | 36.78 | 77.71 | 49.69 | 65.26 | 69.22 | 20.62 | 53.39 |
| 3 bits | 54.27 | 36.94 | 77.58 | 49.90 | 65.11 | 70.09 | 19.48 | 53.34 |
| 4 bits | 54.18 | 36.94 | 77.54 | 49.80 | 64.95 | 69.53 | 19.18 | 53.16 |
| 5 bits | 53.67 | 36.78 | 77.63 | 50.15 | 65.22 | 69.69 | 18.57 | 53.10 |
| 6 bits | 53.67 | 36.85 | 77.64 | 50.20 | 65.07 | 69.69 | 18.80 | 53.13 |
| 7 bits | 53.74 | 37.01 | 77.56 | 50.29 | 65.15 | 69.38 | 18.50 | 53.09 |
| 8 bits | 53.84 | 36.94 | 77.51 | 50.15 | 64.95 | 70.17 | 18.80 | 53.19 |
| <i>Vicuna-7b v1.5</i> | 53.92 | 37.14 | 77.45 | 50.36 | 64.41 | 69.61 | 19.03 | 53.13 |

Table 10: Full results of BitDelta applied to fine-tuned models in the Llama-2 and Mistral families, corresponding to Table 2.

| Model | Method | ARC | BBH | HellaSwag | TruthfulQA | LAMBADA | WinoGrande | GSM8K | Average \uparrow | MT-Bench \uparrow |
|-------------------------------------|------------------|-------|-------|-----------|------------|---------|------------|-------|--------------------|---------------------|
| <i>Llama 2-7B</i> | – | 52.56 | 33.76 | 78.96 | 38.96 | 68.39 | 68.98 | 13.57 | 50.74 | – |
| <i>Llama 2-7B Chat</i> | Baseline | 53.58 | 33.84 | 78.58 | 45.32 | 66.58 | 66.46 | 22.74 | 52.44 | 6.56 |
| | BitDelta-Initial | 55.46 | 35.56 | 76.32 | 41.10 | 68.14 | 68.03 | 18.27 | 51.84 | 6.31 |
| | BitDelta | 54.44 | 33.85 | 78.31 | 44.95 | 66.66 | 66.14 | 20.24 | 52.08 | 6.47 |
| <i>Vicuna-7B v1.5</i> | Baseline | 53.92 | 37.14 | 77.45 | 50.36 | 64.41 | 69.61 | 19.03 | 53.13 | 6.04 |
| | BitDelta-Initial | 54.69 | 36.74 | 78.47 | 47.63 | 66.31 | 68.75 | 19.56 | 53.16 | 5.67 |
| | BitDelta | 54.27 | 36.57 | 77.9 | 49.97 | 65.2 | 69.46 | 20.17 | 53.36 | 5.99 |
| <i>Vicuna-7B v1.5 16k</i> | Baseline | 54.86 | 35.63 | 77.06 | 50.38 | 52.32 | 67.64 | 14.18 | 50.30 | 6.06 |
| | BitDelta-Initial | 55.55 | 33.24 | 77.99 | 45.58 | 56.8 | 68.98 | 13.95 | 50.30 | 5.69 |
| | BitDelta | 54.61 | 34.68 | 77.14 | 48.75 | 53.89 | 67.88 | 14.48 | 50.20 | 6.24 |
| <i>Xwin LM-7B v0.1</i> | Baseline | 57.59 | 34.05 | 79.15 | 48.06 | 68.02 | 69.22 | 10.77 | 52.41 | 6.24 |
| | BitDelta-Initial | 56.40 | 33.90 | 80.26 | 44.56 | 69.86 | 69.14 | 16.68 | 52.97 | 5.79 |
| | BitDelta | 57.94 | 34.19 | 79.36 | 47.62 | 68.29 | 69.53 | 9.02 | 52.28 | 6.50 |
| <i>Llama 2-13B</i> | – | 59.47 | 39.03 | 82.23 | 36.90 | 70.44 | 72.22 | 22.74 | 54.72 | – |
| <i>Llama 2-13B Chat</i> | Baseline | 60.32 | 37.89 | 82.15 | 43.95 | 68.62 | 70.96 | 33.13 | 56.72 | 6.98 |
| | BitDelta-Initial | 59.90 | 38.04 | 82.13 | 41.70 | 69.82 | 71.35 | 33.36 | 56.61 | 7.06 |
| | BitDelta | 59.98 | 38.03 | 81.92 | 43.47 | 68.46 | 71.43 | 31.92 | 56.46 | 6.95 |
| <i>Vicuna-13B v1.5</i> | Baseline | 57.34 | 39.47 | 81.14 | 50.86 | 68.48 | 71.67 | 29.72 | 56.95 | 6.48 |
| | BitDelta-Initial | 54.69 | 36.74 | 78.47 | 47.63 | 66.31 | 68.75 | 31.84 | 54.92 | 6.51 |
| | BitDelta | 57.42 | 39.20 | 81.33 | 50.39 | 68.81 | 71.51 | 30.48 | 57.02 | 6.81 |
| <i>Vicuna-13B v1.5 16k</i> | Baseline | 54.86 | 35.63 | 77.06 | 50.38 | 52.32 | 67.64 | 29.72 | 52.52 | 6.90 |
| | BitDelta-Initial | 59.90 | 38.04 | 82.13 | 41.70 | 69.82 | 71.35 | 26.76 | 55.67 | 6.60 |
| | BitDelta | 54.61 | 34.68 | 77.14 | 48.75 | 53.89 | 67.88 | 28.73 | 52.24 | 6.88 |
| <i>WizardLM-13B v1.2</i> | Baseline | 60.15 | 40.82 | 82.58 | 47.17 | 69.26 | 71.90 | 42.38 | 59.18 | 6.95 |
| | BitDelta-Initial | 60.41 | 40.27 | 83.26 | 44.89 | 70.23 | 71.74 | 42.08 | 58.98 | 6.73 |
| | BitDelta | 60.92 | 41.30 | 82.55 | 46.67 | 68.97 | 71.51 | 41.62 | 59.08 | 6.93 |
| <i>Xwin LM-13B v0.1</i> | Baseline | 63.14 | 40.12 | 82.92 | 45.54 | 70.62 | 73.09 | 21.15 | 56.65 | 6.78 |
| | BitDelta-Initial | 63.4 | 40.33 | 83.71 | 43.6 | 71.26 | 73.09 | 26.76 | 57.45 | 6.70 |
| | BitDelta | 62.80 | 39.81 | 83.01 | 48.19 | 70.74 | 72.30 | 21.76 | 56.94 | 6.83 |
| <i>Llama 2-70B</i> | – | 67.58 | 51.67 | 87.00 | 44.82 | 74.81 | 77.98 | 52.69 | 65.22 | – |
| <i>Llama 2-70B Chat</i> | Baseline | 65.44 | 43.93 | 85.91 | 52.77 | 73.90 | 74.90 | 47.61 | 63.49 | 7.12 |
| | BitDelta-Initial | 63.4 | 38.67 | 81.36 | 41.63 | 72.66 | 73.95 | 42.38 | 59.15 | 6.85 |
| | BitDelta | 65.87 | 44.97 | 85.65 | 51.37 | 74.29 | 74.90 | 48.82 | 63.70 | 7.06 |
| <i>Solar-0-70B</i> | Baseline | 71.16 | 55.54 | 87.78 | 62.03 | 75.04 | 79.32 | 56.18 | 69.58 | 7.07 |
| | BitDelta-Initial | 69.54 | 54.52 | 87.57 | 59.08 | 75.37 | 78.69 | 56.79 | 68.79 | 6.79 |
| | BitDelta | 70.82 | 55.06 | 87.35 | 62.03 | 75.86 | 78.77 | 56.63 | 69.50 | 6.82 |
| <i>Xwin LM-70B v0.1</i> | Baseline | 70.65 | 52.40 | 87.15 | 60.06 | 75.04 | 78.06 | 40.33 | 66.24 | 7.45 |
| | BitDelta-Initial | 69.97 | 52.93 | 87.36 | 60.77 | 75.51 | 78.14 | 50.64 | 67.90 | 7.70 |
| | BitDelta | 70.22 | 52.22 | 86.97 | 58.57 | 75.49 | 77.58 | 40.18 | 65.89 | 7.34 |
| <i>Mistral-7B v0.1</i> | – | 61.35 | 41.18 | 83.46 | 42.60 | 70.10 | 73.80 | 37.76 | 58.61 | – |
| <i>Mistral-7B v0.1 Instruct</i> | Baseline | 55.03 | 38.66 | 75.52 | 55.93 | 63.28 | 69.30 | 32.75 | 55.78 | 6.86 |
| | BitDelta-Initial | 59.22 | 40.25 | 79.91 | 51.27 | 67.63 | 72.14 | 38.82 | 58.46 | 6.54 |
| | BitDelta | 55.38 | 37.95 | 75.62 | 55.23 | 66.06 | 70.48 | 31.54 | 56.04 | 6.43 |
| <i>Zephyr-7B-β</i> | Baseline | 63.82 | 39.04 | 84.33 | 55.12 | 66.23 | 72.69 | 34.34 | 59.37 | 7.18 |
| | BitDelta-Initial | 63.57 | 41.87 | 83.85 | 54.53 | 67.73 | 73.56 | 40.26 | 60.77 | 6.70 |
| | BitDelta | 65.02 | 41.64 | 84.05 | 58.39 | 66.33 | 73.95 | 31.92 | 60.19 | 7.00 |
| <i>OpenChat 3.5</i> | Baseline | 64.51 | 45.28 | 84.39 | 47.34 | 65.19 | 72.61 | 68.84 | 64.02 | 7.74 |
| | BitDelta-Initial | 64.16 | 45.23 | 84.13 | 43.34 | 68.62 | 77.43 | 57.77 | 62.95 | 5.71 |
| | BitDelta | 64.93 | 44.57 | 84.44 | 46.24 | 65.88 | 76.40 | 57.70 | 62.88 | 7.38 |
| <i>Dolphin 2.2.1</i> | Baseline | 64.16 | 44.49 | 83.30 | 54.02 | 69.36 | 75.22 | 54.28 | 63.55 | 7.36 |
| | BitDelta-Initial | 64.16 | 44.43 | 84.01 | 48.14 | 69.98 | 75.30 | 50.27 | 62.33 | 7.10 |
| | BitDelta | 64.59 | 43.08 | 83.44 | 54.91 | 68.39 | 75.37 | 52.84 | 63.23 | 7.20 |
| <i>OpenOrca-7B</i> | Baseline | 62.80 | 44.45 | 83.58 | 52.30 | 66.10 | 73.24 | 50.11 | 61.80 | 6.70 |
| | BitDelta-Initial | 63.74 | 44.46 | 84.15 | 49.66 | 69.05 | 74.03 | 49.96 | 62.15 | 7.12 |
| | BitDelta | 63.65 | 43.46 | 83.49 | 51.67 | 66.12 | 74.27 | 49.58 | 61.75 | 7.05 |

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: The effectiveness of BitDelta is supported by accuracy experiments, performance benchmarks on the kernel and model level, and through ablations like quantizing the base model and repeatedly applying BitDelta.

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: One discussed limitation is how Section 4.3 assumes a toy scenario where all B models simultaneously receive a different request, which is indicative of the worst case scenario of a multi-tenant serving system, but is not necessarily the most representative scenario.

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: The paper does not include theoretical results.

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: The paper fully discusses the BitDelta methodology on an algorithmic level, and how to replicate the experiments in Section 4.

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.

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: [Yes]

Justification: The paper provides the source code to reproduce the main results.

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: The paper provides training details in Section 3.1.

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 paper does not report error bars / statistical significant tests as of now.

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: The authors describe the compute resources they used to apply BitDelta to models of varying sizes in Section 3.1.

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: The authors have reviewed the NeurIPS Code of Ethics and confirm that there are no major ethical concerns with BitDelta.

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: The authors describe both positive and negative societal impacts of BitDelta in Section A.1.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- 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: The paper poses no such risks.

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 have cited the creators/original owners of assets the paper uses/references.

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: [Yes]

Justification: The source code associated with the paper is well documented.

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: The paper does not involve crowdsourcing/research with human subjects.

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/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.
- 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.