

---

# Activation Aware Bitdelta

---

Michael Peng<sup>1\*</sup>


Jessie (Ju Young) Lee<sup>1</sup>

Emily Zhou<sup>1</sup>

Sophia Chen<sup>1</sup>

Shrika Eddula<sup>1</sup>

<sup>1</sup>Massachusetts Institute of Technology

 <https://github.com/mpeng19/Activation-Aware-Bitdelta>

[Link to Demo Video](#)

## Abstract

Quantization techniques have become critical for improving the efficiency of deploying large language models (LLMs) in resource-constrained settings. BitDelta, a method for 1-bit quantization of weight deltas between fine-tuned and base models, demonstrates the feasibility of compressing fine-tuning information while preserving model performance. Despite its efficiency, the original BitDelta framework relies on a computationally expensive distillation step for scale factor refinement, which hinders practical adoption in scenarios with limited resources. In this work, we introduce Activation-Aware BitDelta, an improved scale initialization scheme that incorporates activation statistics to better preserve weight-activation interactions. Our method avoids the costly distillation step by deriving a closed-form solution for scale factors that minimizes activation space error. The proposed method not only simplifies the quantization pipeline and achieves >2x wall-clock time speedups, but also enhances initialization accuracy, demonstrating comparable or improved model performance relative to the original approach. We validate Activation-Aware BitDelta across a suite of benchmarks and model families, showing its capability to maintain compression quality while reducing computational overhead. This contribution opens avenues for more accessible deployment of fine-tuned LLMs, enabling efficient multi-tenant serving without sacrificing quality.

## 1 Introduction

In recent years, the development and deployment of large language models (LLMs) have revolutionized natural language processing (NLP) across academia, industry, and various real-world applications. Models such as GPT, LLaMA, and Mistral have demonstrated impressive capabilities in understanding and generating human-like text. However, the increasing size and complexity of these models have introduced significant challenges in computational efficiency and deployment costs. After pre-training, models are typically fine-tuned for downstream tasks, which improves their performance on a variety of metrics such as Instruction-following and domain-specific skills [9, 19]

BitDelta [8] has emerged as a promising solution for efficient model deployment by compressing the weight differences between fine-tuned and base models to just 1 bit through sign quantization with per-matrix scaling factors. This approach has demonstrated that fine-tuning information can be effectively preserved while drastically reducing storage requirements. However, the current implementation relies on computationally expensive distillation steps for scale factor refinement, which can be prohibitive in resource-constrained environments.

---

\*Michael and Emily worked on setting up the bash scripts and running various experiments, Jessie and Sophia worked on implementing the layer-wise activation statistics collector, Shrika worked on analyzing and graphing experimental results as well as writing a large chunk of the paper report.

To address these challenges, we propose an efficient enhancement to the BitDelta framework that fundamentally changes how scaling factors are initialized and optimized. Rather than relying on expensive distillation procedures, our approach analyzes the statistical properties of activations within the network to determine optimal quantization parameters. By formulating the quantization objective in terms of activation-space error, we derive an analytical solution that requires only a single forward pass to compute appropriate scaling factors. This reformulation not only eliminates the computational bottleneck of iterative distillation but also provides a more principled approach to preserving model behavior during quantization.

## 1.1 Contributions

The key contributions of this work include:

- A novel activation-aware initialization scheme for BitDelta that maintains compression quality while reducing computational overhead
- A closed-form solution for optimal scale factors that incorporates activation statistics
- Comprehensive evaluation across multiple model families and domains, demonstrating comparable or improved performance relative to the original BitDelta approach
- Insights into the trade-offs between computational efficiency and model performance in quantization-based compression

## 2 Related Works

The field of LLM compression and efficient deployment has seen significant developments in recent years. These advances can be broadly categorized into quantization techniques, efficient fine-tuning methods, and activation-aware optimizations.

### 2.1 Quantization Techniques

Efficient training and inference of LLMs have been extensively studied to address the computational challenges posed by models with billions of parameters. The LLaMA series[14] introduced a family of foundational models optimized for efficiency and open accessibility, which have since inspired many derivatives, including Vicuna and Mistral [1, 13]. Techniques like quantization [2] have further reduced memory requirements, enabling the deployment of LLMs in resource-constrained environments such as single-GPU or CPU-only devices [11].

Recent work in model quantization has demonstrated impressive results in reducing model size while preserving performance. GPTQ [5] introduced a one-shot weight quantization method that leverages Hessian information to minimize the quantization error. SmoothQuant [16] proposed a novel quantization scheme that jointly considers activation and weight distributions to enable accurate low-bit quantization. These approaches laid important groundwork for efficient model compression, though they focus primarily on quantizing entire models rather than fine-tuning deltas.

### 2.2 Efficient Fine-tuning

The emergence of parameter-efficient fine-tuning (PEFT) methods has sparked interest in compressing fine-tuning information. LoRA [6] demonstrated that model adaptation could be achieved by learning low-rank updates to pre-trained weights. QLoRA [3] advanced this approach by enabling fine-tuning of 4-bit quantized models while maintaining model quality, though requiring specialized hardware. Various other approaches including BitFit [18] explored other efficient fine-tuning methods such as only fine-tuning the bias weights. BitDelta [8] extended these concepts by showing that fine-tuning information could be effectively compressed to 1-bit representations, though its reliance on distillation for scale refinement presents computational challenges.

### 2.3 Activation-Aware Methods

Activation-aware approaches have shown promise in improving quantization outcomes. AWQ [7] introduced techniques for identifying and preserving weights that have the highest impact on model

outputs based on activation patterns, while SpQR [4] proposed dynamic quantization based on activation statistics. OmniQuant [10] developed a unified framework leveraging both weight and activation distributions for robust quantization. AgileQuant [12] further improved efficiency by introducing fast approximations for activation-based weight importance scoring. These methods demonstrate the value of considering activation patterns in quantization, though they primarily focus on full model quantization rather than fine-tuning deltas. We apply these ideas to the BitDelta framework by finding a closed-form solution for the layer-wise scale factors in terms of the sign matrix and the activation covariance matrix.

### 3 Methodology

In this section, we outline the experimental setup and methodology used to evaluate the impact of activation-aware training on large language models (LLMs) across various architectures and domains. Our approach consists of three primary components: training configurations, model architectures, and evaluation metrics.

#### 3.1 Training Configurations

To evaluate the impact of activation-aware training, we modify the standard BitDelta training pipeline to support the following configurations:

- **Standard Training:** Conventional training without activation awareness, serving as the baseline.
- **Activation-Aware Training:** Incorporating activation-aware loss functions and calibration-specific adjustments. This configuration leverages activation statistics during training to enhance model calibration.

In the activation-aware setting, scale factors  $\alpha$  are optimized to minimize error in the activation space:

$$\alpha^* = \arg \min_{\alpha} \mathbb{E} [\|W_{\text{fine}}X - (W_{\text{base}} + \alpha \odot \text{sign}(\Delta))X\|_2^2],$$

where  $\Sigma = \mathbb{E}[XX^T]$  is the activation covariance matrix, and the closed-form solution is derived as:

$$\alpha^* = \frac{\text{tr}(\Delta \Sigma \text{sign}(\Delta)^T)}{\text{tr}(\text{sign}(\Delta) \Sigma \text{sign}(\Delta)^T)}.$$

The detailed derivation is provided in Appendix A.

#### 3.2 Model Architectures

We evaluate activation-aware training on several 7B-parameter LLMs and their fine-tuned variants. Specifically, we examine the following model pairs:

- **Mistral Pairs:** Base model Mistral-7B-v0.1 paired with its fine-tuned variants Mistral-7B-Instruct-v0.1 and Zephyr-7B-beta.
- **LLaMA-2 Pairs:** Base model LLaMA-2-7B-hf paired with Vicuna-7B-v1.5 and LLaMA-2-7B-chat-hf.

While larger models exist in these families (such as 13B and 70B variants), we focused on 7B parameter models due to GPU memory constraints.

#### 3.3 Evaluation Metrics

To quantify the effectiveness of activation-aware training, we evaluate models on two primary metrics:

- **Perplexity:** A standard metric for measuring language model performance. Lower perplexity indicates better model accuracy.
- **Wall clock time:** The time taken by BitDelta from start to finish (excluding the time taken to load models/datasets and save results). In the standard BitDelta, this includes the time taken to train the scale factors. In our new activation-aware schema, this includes the time it takes to measure the activation statistics and precompute the scale factors.

Evaluations are performed on a set of benchmark datasets representing diverse domains:

- FremyCompany/AGCT-Dataset (Medical Domain)
- bigcode/the\_stack (Code Domain)
- atrost/financial\_phrasebank (Financial Domain)
- wikitext-2-raw-v1 (General Domain)

For each dataset, the perplexity of the recovered fine-tuned model is evaluated on 100 randomly sampled examples from the training split.

### 3.4 Experimental Setup

All experiments are conducted on 2 NVIDIA A100 GPUs with 40GB of memory. Models are distributed across multiple GPUs using balanced device maps where applicable. Memory allocation is optimized using PyTorch’s `max_split_size_mb` configuration, and training scripts are executed with the `transformers` library.

### 3.5 Analysis Pipeline

Post-training, we analyze model performance across configurations by:

- Comparing perplexity scores between standard and activation-aware training for each domain.
- Measuring latency improvements and memory usage for activation-aware models.
- Visualizing results to highlight trade-offs between computational efficiency and accuracy.

## 4 Experiments

In this section, we present the results of our experiments, evaluating the impact of activation-aware training on large language models (LLMs). The experiments are divided into two main parts: (1) wall clock time comparisons and (2) perplexity comparisons across different domains. All models are trained with consistent hyperparameters for a fair comparison.

### 4.1 Wall Clock Time Comparison

Figure 1 shows the end-to-end training time across various batch sizes for standard and activation-aware training. The activation-aware approach is independent of batch size since no training is done, demonstrating the efficiency of leveraging activation statistics during the optimization process.

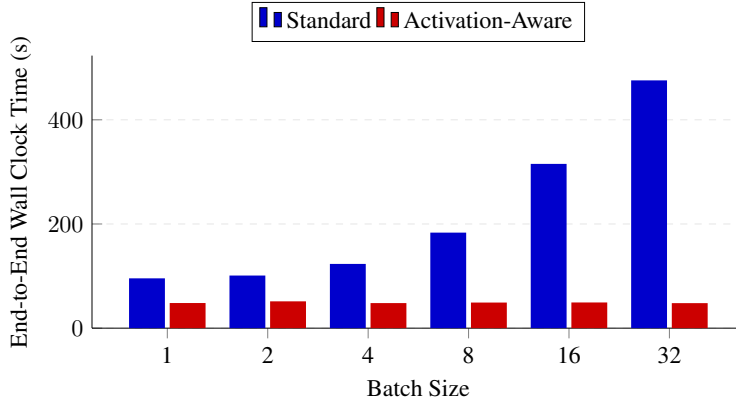


Figure 1: End-to-end wall clock time comparison between standard and activation-aware approaches (using `train_examples = 250`)

## 4.2 Domain-specific results

The results across the four evaluated domains (code, health, finance, and wikitext) demonstrate consistent improvements in perplexity with the activation-aware approach compared to the standard baseline. As shown in Figures 2–5, the activation-aware models achieve lower perplexity across all calibration sizes, with the largest gains observed in structured domains like finance and code. Notably, these improvements are achieved with consistent calibration performance regardless of the calibration size, highlighting the robustness of the activation-aware methodology.

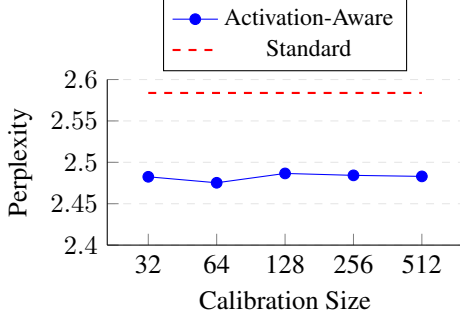


Figure 2: Perplexity comparison for code domain.

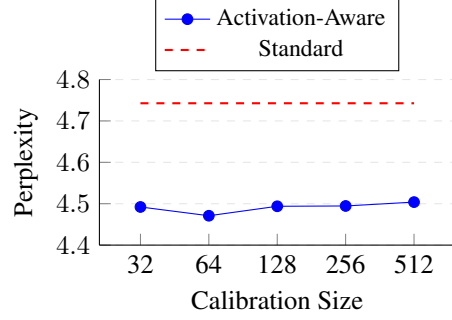


Figure 3: Perplexity comparison for health domain.

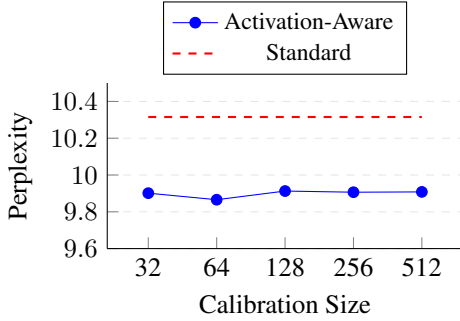


Figure 4: Perplexity comparison for finance domain.

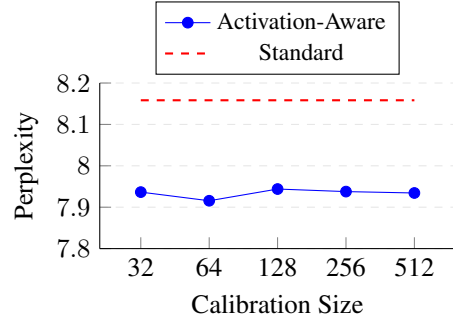


Figure 5: Perplexity comparison for wikitext domain.

## 4.3 Pairwise Comparisons: Perplexity and Latency

We evaluated the impact of activation-aware training on perplexity and latency using model pairs across various configurations. These comparisons highlight the benefits of activation-aware methods for both model performance and computational efficiency.

**Perplexity Comparison (Table 1)** The perplexity results demonstrate that activation-aware configurations consistently outperform the standard approach in most model pairs. Notable improvements ranging from 0.2 – 0.5 perplexity points are observed in all pairs except for *Mistral-7B-v0.1* and *Zephyr-7B-beta*, where activation-aware training yields a perplexity of 8.88 compared to 7.79 for the standard configuration. This outlier may be a result of *zephyr-7B* being fine-tuned with dDPO (distilled direct preference optimization) [15]. In fact, a recent paper [17] suggests that DPO affects the activations in significantly different ways when compared to traditional fine-tuning methods.

**Latency Comparison (Table 2)** Wall clock time experiments further highlight the efficiency of activation-aware methods. Activation-aware models consistently achieve significantly lower wall clock time compared to their standard counterparts, with reductions exceeding 50% in some cases. For instance, the *Llama-2-7B-hf* / *Llama-2-7B-chat-hf* pair shows a dramatic reduction from 146.27 seconds in the standard configuration to 67.56 seconds in the activation-aware setup.

Similar trends are observed across other model pairs, demonstrating the computational advantages of activation-aware training in multi-tenant serving scenarios.

Model Pair	Activation-Aware	Standard
Mistral-7B-v0.1 / Zephyr-7B-beta	8.878828	7.790850
Mistral-7B-v0.1 / Mistral-7B-Instruct-v0.1	7.864079	8.368696
Llama-2-7B-hf / Vicuna-7B-v1.5	7.581587	7.841721
Llama-2-7B-hf / Llama-2-7B-chat-hf	7.937717	8.158337

Table 1: Perplexity comparison for activation-aware and standard configurations (lower is better).

Model Pair	Activation-Aware	Standard
Mistral-7B-v0.1 / Zephyr-7B-beta	77.36	150.73
Mistral-7B-v0.1 / Mistral-7B-Instruct-v0.1	92.04	143.06
Llama-2-7B-hf / Vicuna-7B-v1.5	77.32	141.51
Llama-2-7B-hf / Llama-2-7B-chat-hf	67.56	146.27

Table 2: Wall clock time comparison (in seconds) for activation-aware and standard configurations.

## 5 Conclusion

In this work, we demonstrated the effectiveness of activation-aware training in improving both performance and efficiency for large language models. Through domain-specific evaluations and pairwise comparisons, we showed that activation-aware methods consistently achieve lower perplexity and significantly reduce latency compared to standard configurations. For instance, in the Llama-2-7B-hf / Llama-2-7B-chat-hf pair, activation-aware training reduced latency by more than 50% while maintaining superior perplexity performance. These results highlight the potential of activation-aware techniques to enhance model calibration and scalability, making them highly applicable for real-world, multi-tenant deployments.

Future work could explore extending activation-aware techniques to larger model architectures, such as Mistral-Large or LLaMA-3-13B/Llama-3-70B, to further validate their scalability. Additionally, incorporating dynamic calibration techniques that adapt activation-aware scales during inference could enhance performance in settings with dynamic input distributions. Another promising direction is integrating activation-aware training with structured pruning or low-rank approximation methods to further reduce memory overhead while maintaining accuracy. Finally, investigating the impact of activation-aware training on downstream tasks, such as question answering or summarization, could provide insights into its applicability across a wider range of real-world applications.

## References

- [1] Mistral AI. Mistral 7b: A next-generation dense language model. *arXiv preprint arXiv:2310.06801*, 2023.
- [2] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.
- [3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.
- [4] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefer, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression, 2023.
- [5] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.
- [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- [7] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024.
- [8] James Liu, Guangxuan Xiao, Kai Li, Jason D. Lee, Song Han, Tri Dao, and Tianle Cai. Bitdelta: Your fine-tune may only be worth one bit, 2024.
- [9] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.
- [10] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models, 2024.
- [11] Haihao Shen, Hanwen Chang, Bo Dong, Yu Luo, and Hengyu Meng. Efficient llm inference on cpus, 2023.
- [12] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. Agile-quant: Activation-guided quantization for faster inference of llms on the edge, 2023.
- [13] Vicuna Team. Vicuna: An open-source chatbot based on llama models. *GitHub Repository*, 2023. Available at <https://github.com/lm-sys/Vicuna>.
- [14] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [15] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Cl  mentine Fourier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of lm alignment, 2023.
- [16] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024.
- [17] Yushi Yang, Filip Sondej, Harry Mayne, and Adam Mahdi. Ablation is not enough to emulate dpo: How neuron dynamics drive toxicity reduction, 2024.
- [18] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022.
- [19] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

## A Derivation of Closed-Form Solution for Optimal Scale Factors

We aim to minimize the activation error in the objective:

$$\alpha^* = \arg \min_{\alpha} \mathbb{E} [\|W_{\text{fine}}X - (W_{\text{base}} + \alpha \odot \text{sign}(\Delta))X\|_2^2].$$

Substituting  $W_{\text{fine}} = W_{\text{base}} + \Delta$  and simplifying, the problem reduces to:

$$\alpha^* = \arg \min_{\alpha} \mathbb{E} [\|(\Delta - \alpha \odot \text{sign}(\Delta))X\|_2^2].$$

Expanding the squared norm:

$$\|(\Delta - \alpha \odot \text{sign}(\Delta))X\|_2^2 = \text{tr}((\Delta - \alpha \odot \text{sign}(\Delta))\Sigma(\Delta - \alpha \odot \text{sign}(\Delta))^T),$$

where  $\Sigma = \mathbb{E}[XX^T]$  is the activation covariance matrix.

Denoting  $A = \Delta - \alpha \odot \text{sign}(\Delta)$ , the objective becomes:

$$\text{tr}(A\Sigma A^T) = \text{tr}(\Delta\Sigma\Delta^T) - 2\alpha\text{tr}(\Delta\Sigma\text{sign}(\Delta)^T) + \alpha^2\text{tr}(\text{sign}(\Delta)\Sigma\text{sign}(\Delta)^T).$$

Taking the derivative with respect to  $\alpha$  and setting it to zero:

$$\frac{\partial}{\partial \alpha} [\text{tr}(\Delta\Sigma\Delta^T) - 2\alpha\text{tr}(\Delta\Sigma\text{sign}(\Delta)^T) + \alpha^2\text{tr}(\text{sign}(\Delta)\Sigma\text{sign}(\Delta)^T)] = 0,$$

we obtain:

$$\alpha^* = \frac{\text{tr}(\Delta\Sigma\text{sign}(\Delta)^T)}{\text{tr}(\text{sign}(\Delta)\Sigma\text{sign}(\Delta)^T)}.$$