BitDP: Ultra-Low Bit Communication for Efficient Data Parallelism in LLM Training

Anonymous ACL submission

Abstract

Large language model (LLM) training demands extensive data parallelism, resulting in massive gradient communication overhead. While gradient quantization presents a promising solution, it faces two critical challenges: maintaining training stability for transformer architectures and adapting to modern AllReduce-based distributed communication systems. In this paper, we propose BitDP, an ultra-low bit gradient quantization and data parallelism system that reduces communication costs by up to 32× while preserving model accuracy with less than 1% performance degradation. Our approach ensures numerical stability for large transformer models and seamlessly integrates with existing AllReduce infrastructures. We validate BitDP's effectiveness across various LLM sizes and architectural variants, achieving significant communication efficiency improvements while maintaining convergence quality. These results establish BitDP as a scalable and reliable solution for real-world LLM training at industrial scales.

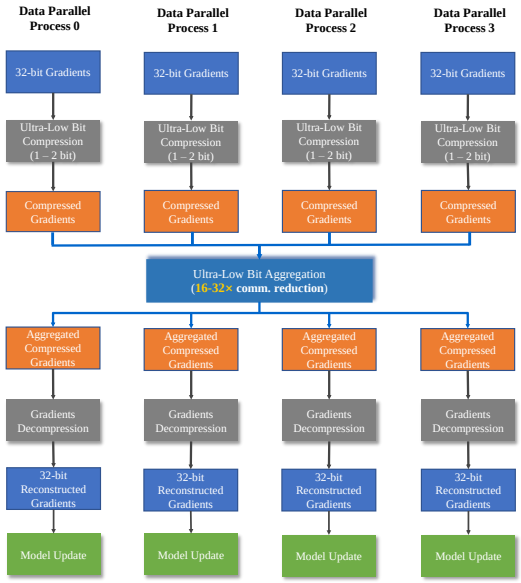


Figure 1: Workflow of BitDP (Ultra-Low Bit Data Parallel) training. The proposed system reduces communication bandwidth by up to 32× through gradient quantization and low-bit AllReduce operations while preserving large language model training accuracy across distributed nodes.

1 Introduction

Large Language Models (LLM) such as GPT-3 (Brown et al., 2020), GPT-4 (Achiam et al., 2023), Gemini (Team et al., 2023), and DeepSeek-R1 (Guo et al., 2025) have been making rapid advances. These models, built on scalable transformer architectures, have achieved remarkable performance in tasks such as natural language processing and multimodal generation. However, their immense scale introduces significant technical challenges, particularly in distributed training, where communication is the performance bottleneck (Rajbhandari et al., 2020; Narayanan et al., 2021).

A major challenge in distributed LLM training lies in the exchange of gradients during backpropagation (Wen et al., 2017; Alistarh et al., 2017; Wang et al., 2024). As model sizes grow, the volume of gradient data increases exponentially, leading to heavy network bandwidth demands and sub-optimal utilization of computational resources. This problem is particularly pronounced at scale, where communication overhead often dominates computation time.

To address this issue, gradient compression techniques such as quantization and sparsification have been proposed (Wen et al., 2017; Alistarh et al., 2017; Wang et al., 2024; Lin et al., 2018; Shi et al., 2021). While effective for simpler models like CNNs, these methods have not been widely applied to LLMs due to limited compression rates (e.g., 8-bit quantization) or concerns about training stability. Achieving ultra-low bit compression, such as 1-bit or 2-bit gradients, remains a significant challenge, as it often introduces noise that destabilizes

training, particularly with optimizers like Adam (Reddi et al., 2018). Moreover, current communication libraries such as the NVIDIA Collective Communications Library (NCCL) (NVIDIA Corporation, 2020) lack native support for such low-bit operations, further complicating their practical deployment.

In this paper, we introduce **BitDP**, a novel system for ultra-low bit gradient communication in LLM training. BitDP achieves up to a 32× reduction in communication overhead by employing novel 1-bit and 2-bit gradient quantization, coupled with a specialized 1-bit AllReduce algorithm that ensures training stability and compatibility with existing communication interfaces. As shown in Figure 1, BitDP seamlessly integrates into distributed training pipelines, enabling efficient large-scale LLM training while maintaining comparable model performance to full-precision approaches. The main contributions of this paper are as follows:

- We propose **BitDP**, a comprehensive system for efficient LLM training that incorporates ultra-low bit (1-bit and 2-bit) gradient quantization techniques, reducing communication overhead by up to 32× while maintaining comparable model performance.
- We develop a novel 1-bit AllReduce algorithm as a core component of BitDP to support ultra-low bit communication, leveraging existing communication frameworks for practical deployment while preserving computational and memory efficiency.
- We validate the entire BitDP system with extensive experiments on various LLMs, demonstrating its scalability, robustness, and effectiveness across different model sizes, architectures, and training configurations without compromising convergence behavior.

2 Related Work

In this section, we review the existing literature on gradient quantization, gradient sparsification, and low bit optimizer, which are closely related to our work.

Gradient Quantization Gradient quantization techniques reduce communication overhead by encoding gradients using fewer bits. Early approaches like QSGD (Alistarh et al., 2017) and TernGrad (Wen et al., 2017) showed promise for vision tasks but were primarily designed for parameter server architectures, not the AllReduce-based systems essential for modern LLM training. Additionally, these methods lacked validation on transformer architectures, where numerical stability presents unique challenges (Chowdhery et al., 2023).

Recent LLM-specific quantization methods such as FP8-LM (Peng et al., 2023) and INT4 approaches (Xi et al., 2023; Wang et al., 2024) achieve results comparable to full-precision training, but only reduce to 4-8 bits per gradient element. In contrast, BitDP pushes boundaries to ultra-low 1-2 bit communication while maintaining training stability and integrating with modern collective communication interfaces, representing a substantial advancement over existing approaches.

Gradient Sparsification Gradient sparsification reduces communication overhead by transmitting only the most significant gradient elements (Lin et al., 2018; Shi et al., 2021). These methods prioritize critical gradient information to maintain convergence while decreasing bandwidth requirements. However, despite these benefits, the selection and tracking of significant gradient elements often requires complex thresholding mechanisms that add computational overhead. Furthermore, these techniques lack validation on large-scale LLM training and integration with modern distributed training infrastructures, limiting their practical applicability for today's most demanding model training scenarios.

Ultra-low Bit Optimizer Unlike our approach that compresses gradients, several studies have investigated the use of 1-bit quantization for storing and exchanging optimizer states in adaptive learning rate optimizers (Tang et al., 2021; Li et al., 2022; Lu et al., 2023). However, these techniques typically cannot train with 1-bit quantization directly from scratch. The extreme compression to 1-bit representation introduces significant quantization errors that need careful management to avoid compromising model convergence. To address this issue, these methods require additional error compensation computations to maintain training accuracy, which typically introduces extra storage requirements. Most critically, these methods often cooperate with parameter server architectures rather than modern AllReduce-based distributed training systems that have become standard for efficient LLM training at scale.

3 Method

An effective ultra-low bit gradient compression technique should simultaneously address two key challenges: minimizing accuracy degradation to maintain model performance, and enabling efficient ultra-low bit AllReduce operations to reduce communication overhead. These challenges are particularly pronounced in distributed LLM training, where gradient synchronization across workers often dominates the overall training time.

In this section, we introduce the two key components of our approach to address the aforementioned challenges. First, we present our Fine-Grained gradient quantization algorithm, which enables precise control of gradient data at ultra-low bit levels, selectively compressing communication-intensive gradients while preserving critical information. Second, we describe our 1-bit AllReduce method, which leverages advanced collective communication primitives and hardware capabilities to significantly reduce gradient synchronization overhead. Together, these components achieve a reduction factor of 16 to 32 times in communication volume without compromising training accuracy.

3.1 Fine-grained Ultra-low Bit Gradient Quantization

3.1.1 Selective Compression for High-volume Gradient Segments

The parameter gradient volumes for various components of a standard GPT model are summarized in Table 1. In large language model (LLM) training, most of the gradient communication volume comes from QKV weights, linear weights, and MLP weights, which scale quadratically with the hidden dimension. In contrast, smaller components such as biases and layer normalization contribute minimally. For instance, in GPT-3, the four largest dense weights account for 99.6% of the total gradient volume. In LLaMA (Touvron et al., 2023), this proportion is even higher due to optimizations like rotary position embedding (Su et al., 2024) and RMSNorm (Zhang and Sennrich, 2019).

In our proposed gradient quantization strategy, we control the gradient precision of different parameters with fine granularity. More specifically, we quantize the gradient of all linear weight to ultra-low bit, such as 2-bit or 1-bit, while maintaining the original gradient precision for other components.

Parameter Name	Gradient Volume
`Word Embeddings`	vd
`Position Embeddings`	sd
`Layernorm Weight`	$(2n+1)d$
`Layernorm Bias`	$(2n+1)d$
`ATT QKV weight`	$3nd^2$
`ATT QKV Bias`	$3nd$
`ATT Linear weight`	nd^2
`ATT Linear Bias`	nd
`MLP h_4h Weight`	$4nd^2$
`MLP h_4h Bias`	$4nd$
`MLP 4h_h Weight`	$4nd^2$
`MLP 4h_h Bias`	nd

Table 1: Gradient communication volume per data parallel in standard GPT model training. n is the number of layers, d is the hidden dimension, v is the vocabulary size, and s is the sequence length.

3.1.2 Channel-adaptive Ultra-low Bit Gradient Quantization

A generalized gradient quantization algorithm f_{quant} maps the full-precision gradient $\mathbf{g}$ to a scaling factor $\mathbf{s}$ and a low-precision quantized gradient $\mathbf{q}$:

$$(\mathbf{s}, \mathbf{q}) = f_{\text{quant}}(\mathbf{g}) \tag{1}$$

This process preserves essential gradient information while reducing precision for efficient communication.

Ultra-low bit quantization can be achieved using binary quantization ($\mathbf{q}_i \in \{-1, 1\}$) or ternary quantization ($\mathbf{q}_i \in \{-1, 0, 1\}$). Traditional methods use a single scaling factor for the entire gradient tensor (Peng et al., 2023; Wen et al., 2017) or eliminate scaling factors altogether (Bernstein et al., 2018), but both approaches hinder the adaptive learning rate mechanism of ADAM.

Unlike SGD, which uses a shared learning rate for all parameters, ADAM employs parameter-wise adaptive learning rates based on the first and second moments of the gradients (Kingma and Ba, 2015):

$$\begin{aligned} \mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\ \mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\ \theta_{t+1} &= \theta_t - \mathbf{m}_t \frac{\alpha}{\sqrt{\mathbf{v}_t + \epsilon}} \end{aligned} \tag{2}$$

This adaptivity is sensitive to scaling factor design in gradient quantization.

In binary quantization, using a single scaling factor $\mathbf{s}_t$ causes the adaptive learning rate $\frac{\alpha}{\sqrt{\mathbf{v}_t+\epsilon}}$ to become constant, as $\mathbf{v}_t$ (the moving average of

$\mathbf{g}_t^2$) is uniform across all parameters. Similarly, in ternary quantization, $\mathbf{g}_t^2$ can only take two values ($\mathbf{s}_t^2$ or 0), further limiting adaptivity.

To address this issue, we propose channel-adaptive ultra-low bit quantization, which applies channel-wise scaling factors instead of a single tensor-wide factor. This approach achieves high compression rates while preserving the adaptivity of ADAM's learning rate.

For a full precision gradient tensor $\mathbf{g} \in \mathbb{R}^{m \times n}$, the goal is to approximate it as $\mathbf{g} \approx \mathbf{s} \odot \mathbf{q}$, where $\mathbf{s} \in \mathbb{R}^m$ is a channel-wise scaling vector and $\mathbf{q} \in \mathbb{I}^{m \times n}$ is a quantized tensor, $\odot$ denotes the Hadamard product. The elements of $\mathbf{q}$ are constrained to $\{-1, 1\}$ for 1-bit quantization and $\{-1, 0, 1\}$ for 2-bit quantization.

To minimize the quantization error, we solve the following optimization problem:

$$\min_{\mathbf{s},\mathbf{q}} L(\mathbf{s}, \mathbf{q}) = \|\mathbf{g} - \mathbf{s} \odot \mathbf{q}\|_2^2 \tag{3}$$

This objective reduces the squared Euclidean distance between the original gradient $\mathbf{g}$ and its quantized approximation.

This optimization can be solved independently for each channel i. The optimal scaling factor s_i^* is derived by minimizing L with respect to s_i:

$$s_i^* = \frac{1}{|\Omega_{\alpha_i}|} \sum_{j \in \Omega_{\alpha_i}} |g_{i,j}| \tag{4}$$

Here, Ω_{α_i} is the set of indices where $|g_{i,j}| > 0$ for channel i, and $|\Omega_{\alpha_i}|$ represents the number of non-zero elements.

For 1-bit quantization, the quantized tensor $\mathbf{q}$ is defined as:

$$q_{i,j} = \begin{cases} 1, & \text{if } g_{i,j} \geq 0 \\ -1, & \text{if } g_{i,j} < 0 \end{cases} \tag{5}$$

The corresponding scaling factor simplifies to the mean of absolute values:

$$s_i^* = \frac{1}{n} \sum_{j=1}^{n} |g_{i,j}| \tag{6}$$

For the 2-bit variant, the quantization function $\mathbf{q}$ introduces sparsity by mapping small gradient values to zero. It is defined as:

$$q_{i,j} = \begin{cases} 1, & \text{if } g_{i,j} \geq \alpha_i \\ 0, & \text{if } -\alpha_i \leq g_{i,j} < \alpha_i \\ -1, & \text{if } g_{i,j} < -\alpha_i \end{cases} \tag{7}$$

Algorithm 1: BitDP: Ultra-low Bit Data Parallel Training Algorithm.

Input: Gradient tensors of all N modules in model $g = \{g^0, ..., g^N\}$, quantization tensor list $List$, bit mode $b \in \{1, 2\}$, threshold factor $\beta = \frac{3}{4}$, data parallel group size P

```
for name, g^k in g do
    if name in List then
        for channel i in g^k do
            if b == 1 then
                q_i^k = Q_1bit(g_i^k)
                s_i^k = (Σ_j |g_{i,j}|) / n
            else
                α_i = β E_j[|g_{i,j}|]
                q_i^k = Q_2bit(g_i^k, α_i)
                s_i^k = (Σ_j |g_{i,j}|) / (Σ_j |q_{i,j}|)
            end
        end
        g_ar^k = ULB-AllReduce(g^k, s^k, P)
         // Ultra-low Bit AllReduce
    else
        g_ar^k = AllReduce(g^k)/P
         // Standard AllReduce
    end
end
```

where α_i is a channel-specific threshold. This threshold determines the sparsity level, and when $\alpha_i = 0$, Equation (7) reduces to the 1-bit quantization function in Equation (5).

The optimal scaling factor $\mathbf{s}$ is obtained by substituting Equation (4) into Equation (3). The threshold α_i can be optimized by maximizing the following objective:

$$\max_{\alpha_i} \frac{1}{|\Omega_{\alpha_i}|} \left(\sum_{j \in \Omega_{\alpha_i}} |g_{i,j}| \right)^2 \tag{8}$$

where Ω_{α_i} denotes the set of indices where $|g_{i,j}| > \alpha_i$.

Since Equation (8) has no analytical solution, we approximate the optimal threshold α_i as $\alpha_i^* = \beta_i \max_j(|g_{i,j}|)$, where β_i is a coefficient based on the gradient distribution. Through experimental observations, we found that gradient magnitudes approximately follow a symmetric triangular distribution centered at zero (see Appendix A for visualization). Under this observation, β_i is derived

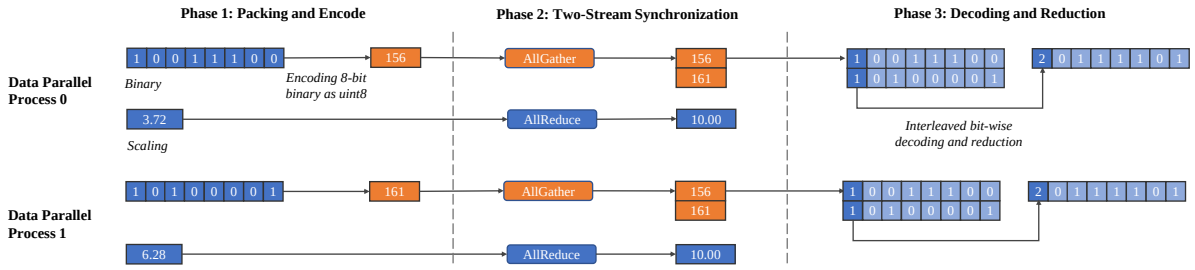


Figure 2: Illustration of Ultra-Low Bit AllReduce. In Phase 1, quantized gradients (shown here with 1-bit quantization for clarity) along with their corresponding scaling factors are processed, with every 8 binary bits packed into a single UINT8 value to optimize communication efficiency. Phase 2 performs parallel communication where scaling factors are synchronized using full-precision AllReduce operations while the compressed gradient matrices use UINT8-based AllGather operations. Phase 3 implements an interleaved bit-wise decoding and reduction strategy that efficiently reconstructs and aggregates the gradients without requiring excessive intermediate memory for complete decompression.

as:

$$\alpha_i^* = \frac{h_i}{4} = \frac{3}{4}\mathbb{E}_j[|g_{i,j}|] \tag{9}$$

where $h_i = \max_j(|g_{i,j}|)$ is the maximum gradient magnitude in channel i, and $\mathbb{E}_j[|g_{i,j}|]$ is the mean absolute gradient value. We use $\mathbb{E}_j[|g_{i,j}|]$ in practice to reduce sensitivity to outliers. A detailed mathematical proof is provided in Appendix B.

Based on the ultra-low bit gradient quantization scheme described above, we further propose a novel ultra-low bit AllReduce communication algorithm that transforms our theoretical communication savings into practical reality. While the detailed communication mechanism will be elaborated in the next subsection, we present the complete BitDP algorithm flow in Algorithm 1, which integrates both the fine-grained ultra-low bit gradient quantization and the ultra-low bit AllReduce procedure. Specifically, the quantization tensor list `List` includes all linear layers in the model, excluding biases, embedding layers, and normalization layers, ensuring that the most communication-intensive components are effectively compressed.

3.2 Ultra-low Bit AllReduce

In distributed LLM training, gradient synchronization typically uses FP32 AllReduce operations to prevent numerical overflow. However, existing frameworks like NCCL lack native support for ultra-low precision gradients (1-bit or 2-bit), creating inefficiencies in communication for such formats.

Quantized gradients consist of a channel-wise FP32 scaling vector and ultra-low precision (1-bit or 2-bit) tensors. While FP32 vectors can be synchronized using standard AllReduce, synchronizing quantized tensors is more challenging. Representing low-precision values as INT8 formats is inefficient, wasting 75% of storage space for 2-bit gradients and risking overflow when the number of workers exceeds 127.

To overcome these challenges, we propose specialized ultra-low bit communication algorithms for 1-bit and 2-bit gradients. Key techniques include bit-level gradient packing, efficient AllGather-based aggregation, two-stream synchronization mechanism, and interleaved bit-wise decoding and reduction. These methods fully exploit the bandwidth savings of ultra-low bit quantization while enabling efficient gradient aggregation. The overall workflow of the proposed methods is illustrated in Figure 2.

- **Bit-level Packing and Encoding** For 1-bit quantization, gradient values are first mapped from $\{-1, 1\}$ to $\{0, 1\}$ for efficient storage. These binary values are grouped into buckets of 8 and packed into a single UINT8 byte using bit-wise encoding.

  For 2-bit quantization, we separate the original gradient into two matrices: one retaining -1 values and another retaining 1 values, setting all other elements to 0. The -1 matrix is mapped from $\{-1, 0\}$ to $\{1, 0\}$. Both matrices are then packed into UINT8 integers using the same bit-wise strategy as in the 1-bit case.

- **AllGather Based Gradient Aggregation** To prevent overflow, we divide AllReduce into two steps: AllGather followed by local reduction. During AllGather, each data-parallel unit

exchanges encoded gradients with all other ranks using efficient UINT8 MPI primitives. Since no arithmetic operations occur in this phase, overflow is avoided and all units receive a complete copy of the encoded gradients for subsequent reduction.

- **Two-Stream Synchronization** We synchronize the quantized gradients and scaling vectors in parallel. Quantized gradients are aggregated using the AllGather process, while scaling vectors are synchronized via AllReduce. The final gradient is computed as:

$$g_{\mathsf{ar}} = \frac{1}{P} \odot s_{\mathsf{ar}} \odot q_{\mathsf{ar}} \tag{10}$$

where P is the data-parallel group size, s_{ar} is the reduced scaling vector, and q_{ar} is the reduced quantized gradient.

- **Interleaved Bit-Wise Decoding and Reduction** During reduction, we interleave the decoding and summation processes at the bit level. For 1-bit quantization, each bit is immediately decoded to $\{-1, 1\}$ and summed across the corresponding positions from all gathered tensors. For 2-bit quantization, we sequentially decode and reduce each bit position from the two separate matrices containing -1s and 1s. This interleaved approach processes one bit at a time, eliminating the need to store fully decoded gradients in memory and significantly reducing peak memory.

With combination of fine-grained ultra-low bit gradient quantization and efficient AllReduce implementation, our method effectively reduces communication overhead while maintaining training accuracy. In the following section, we present comprehensive experimental results to demonstrate the effectiveness of our approach across various large language model training scenarios.

4 Experiments

In this section, we present a comprehensive experimental evaluation of our fine-grained ultra-low-bit gradient quantization algorithm. First, we demonstrate that the proposed algorithm achieves superior performance compared to fully quantized methods and other gradient quantization techniques, such as QSGD and TernGrad. Next, we validate the scalability of our algorithm by evaluating it across varying model sizes and types. Finally, we evaluate the communication efficiency of the system. All experiments are conducted based on PyTorch (Paszke et al., 2019) and Megatron-LM (Shoeybi et al., 2019) framework.

Method	Training Loss	Validation PPL
Baseline	3.086	22.47
FG-2bit	3.118	23.19
FG w/LN	3.120	23.22
Full-2bit	Diverge	Diverge
FG w/Embd	Diverge	Diverge
FG w/bias	Diverge	Diverge

Table 2: Granularity Strategy Comparison for GPT-350M Model

4.1 Performance Evaluation of Fine-Grained Ultra-Low Bit Quantization

We validated the effectiveness of our fine-grained algorithm using a GPT-3 medium-sized model with 350 million parameters. The model, based on the standard Transformer decoder architecture, was trained on a mixed dataset of BooksCorpus (Zhu et al., 2015), English Wikipedia (Foundation), and OpenWebText (Gokaslan and Cohen, 2019), distributed across 8 GPUs. A detailed dataset information is provided in Appendix C.

All experiments were conducted with a global batch size of $65,536$ tokens. The AdamW optimizer was used with a learning rate of $3e-4$ and a weight decay of 0.01. Hyperparameters and settings were kept consistent across all evaluations.

Table 2 summarizes the results of various granularity strategies applied to the GPT-350M model. The fine-grained 2-bit quantization method (FG-2bit) shows minimal degradation in training loss and validation perplexity (PPL) compared to the baseline, which uses FP32 precision gradients.

In contrast, full 2-bit quantization across all modules, as well as FG-2bit with embedding gradients (FG w/Embd) or bias gradients (FG w/Bias), results in divergence during training. Incorporating layer normalization (FG w/LN) demonstrates that layer normalization is compatible with fine-grained 2-bit quantization, achieving comparable results to FG-2bit in terms of training loss and validation perplexity. Overall, FG-2bit remains the preferred approach due to its optimal balance between performance and efficiency.

As shown in Figure 3, we compared the FG-2bit algorithm with other popular quantization methods, including TernGrad and QSGD (in ternary

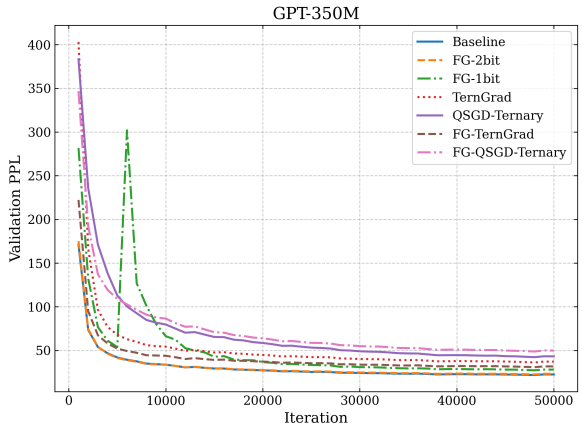


Figure 3: Comparison of training loss across different gradient quantization algorithms.

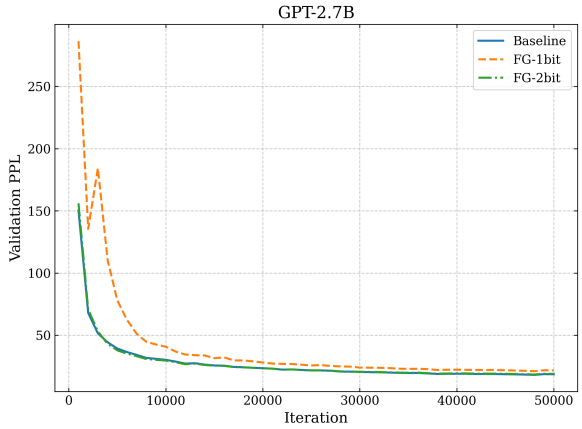


Figure 4: Comparison of training loss across different gradient quantization algorithms.

version for consistency). Fine-grained versions of these methods, namely FG-TernGrad and FG-QSGD-Ternary, were evaluated to align with our approach and are explicitly labeled in the figure for clarity. The results show that FG-2bit significantly outperforms all competing methods in validation perplexity (PPL), demonstrating its robustness in maintaining model performance under reduced precision. Additionally, we tested a fine-grained 1-bit variant that uses the gradient sign. While it showed numerical instability during pretraining and underperformed FG-2bit, it still exceeded the performance of all other 2-bit methods.

4.2 Scalability and Generalization Evaluation

4.2.1 Scalablity

To evaluate the scalability of our algorithms, we tested the pre-training performance of a GPT model with 2.7 billion parameters, as shown in Figure 4. This model shares the same architecture as the 350M model but features an expanded dimension of 2560, 32 layers, and 32 attention heads, consistent with the GPT-3 2.7B specification. The experiments were conducted on 32 GPUs distributed across 4 servers, using the same dataset and hyperparameter settings as the 350M model, except for a peak learning rate adjusted to 2×10^{-4}. All configurations were kept consistent to ensure reliable scalability comparisons.

Method	Training Loss	Validation PPL
Baseline	2.895	18.60
FG-2bit	2.913	18.92
FG-1bit	3.052	21.25

Table 3: Comparison for GPT-2.7B Model

As shown in Table 3, the FG-2bit model achieves performance close to the baseline full-precision model, with only a 0.018 difference in training loss and 0.32 in validation perplexity (PPL). Notably, this gap is smaller than that observed in the 350M model, where the differences were 0.032 in training loss and 0.72 in PPL. These results highlight the scalability of the FG-2bit approach, as it maintains high performance even for larger models.

Additionally, we assessed the performance of a 1-bit quantization for the 2.7B model. Similar to the 350M model, the 1-bit version initially exhibited instability in training loss. However, as shown in Figure 4 and detailed in Table 3, it eventually converged to a level close to that of the baseline, demonstrating its potential viability despite the early-stage fluctuations.

4.2.2 Generalization to LLaMA

To validate the generalization capability of our ultra-low bit quantization method across different architectures and larger models, we conducted experiments on LLaMA, an 8B parameter model with a distinctly different architecture from GPT. The model features 32 layers, 4096 hidden dimensions, and five key architectural differences from GPT: (1) RMSNorm instead of LayerNorm for normalization, (2) Rotary Positional Embedding (RoPE) instead of learned positional embeddings, (3) SwiGLU (Shazeer, 2020) activation functions instead of GELU, (4) Grouped-Query Attention (GQA) (Ainslie et al., 2023) instead of standard multi-head attention, and (5) the absence of bias terms in linear layers. We trained each model configuration for 50,000 steps with 65,536 tokens per batch. The dataset composition remained consis-

Method	ARC-E	BoolQ	H-Swag	PIQA	COPA	REC	Avg
Baseline	47.3	56.7	31.3	65.2	69.0	73.0	57.1
FG-2bit	47.6	57.0	30.6	63.8	72.0	71.6	57.1
FG-1bit	47.3	55.9	30.4	63.9	72.0	71.0	56.8

Table 4: Performance evaluation of LLaMA-8B on various benchmarks (accuracy in %). Baseline uses FP32 gradient, while FG-2bit and FG-1bit represent our proposed fine-grained 2-bit and 1-bit fine-grained quantization methods respectively. Benchmarks include ARC-Easy (ARC-E) (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (H-Swag) (Zellers et al., 2019), Physical Interaction QA (PIQA) (Bisk et al., 2020), Choice of Plausible Alternatives (COPA) (Roemmele et al., 2011), and ReCORD (REC) (Zhang et al., 2018).

tent with our previous experiments on 350M and 2.6B parameter GPT models.

Our evaluation across multiple benchmarks yielded promising results. As shown in Table 4, our fine-grained 2-bit quantization maintains the baseline's 57.1% average performance across all tasks. Even more remarkably, our 1-bit quantization achieved 56.8% average performance, with only a 0.3% degradation from full precision.

This exceptional quantization efficiency on LLaMA models can be attributed to their bias-free architecture. Our experiments with smaller models demonstrated that quantizing bias gradients often causes training instability and divergence. LLaMA's elimination of bias terms in linear layers removes this quantization-sensitive component, significantly narrowing the performance gap between ultra-low-bit and full-precision training. This finding reinforces the importance of considering architectural decisions that can facilitate more efficient gradient quantization for large-scale training.

Model	DP	Baseline	FG2	FG1
350M	8	27.96	26.17	18.24
350M	16	183.38	125.71	82.82
350M	32	246.79	258.86	162.55
1.3B	8	75.63	40.09	27.33
1.3B	16	595.64	540.61	231.82
1.3B	32	859.22	1059.80	493.04
2.7B	4	158.23	39.21	24.17
2.7B	8	1029.02	242.87	132.18
2.7B	16	1332.90	525.93	276.46
6.7B	2	178.66	27.54	18.68
6.7B	4	3813.32	248.88	136.48
6.7B	8	4974.62	692.81	361.99
13B	2	6494.37	153.45	70.33
13B	3	8217.07	277.49	133.16
13B	4	9225.83	555.43	241.68

Table 5: All-reduce communication time (ms) for different model sizes and data parallelism (DP) configurations. FG2 and FG1 represent our proposed fine-grained 2-bit and 1-bit quantization methods respectively.

4.3 Communication Efficiency Analysis

Table 5 presents the communication time measurements for gradient all-reduce operations across various model sizes and data parallelism configurations. Our fine-grained ultra-low bit quantization methods (FG2 and FG1) demonstrate substantial communication time reductions compared to the FP32 baseline. The benefits become increasingly pronounced as model size and parallelism degree increase. For the 13B parameter model with 4-way data parallelism, our 2-bit quantization reduces communication time from 9225.83ms to just 555.43ms (16.6× speedup), while our 1-bit quantization further reduces it to 241.68ms (38.2× speedup). Significant improvements are also observed with the 6.7B model at 4-way parallelism, where 2-bit quantization achieves a 15.3× speedup and 1-bit quantization achieves a 27.9× speedup. These results confirm that our ultra-low bit quantization techniques effectively address the communication bottleneck in distributed training of large language models, with particularly significant benefits in clusters with limited network bandwidth where communication overhead is most severe.

5 Conclusion

In this paper, we introduced BitDP, an ultra-low bit gradient quantization and data parallelism system designed to address the massive communication overhead in LLM training. Our approach reduces communication costs by up to 32× with less than 1% performance degradation. Experiments demonstrate BitDP's scalability and reliability as a practical solution for large-scale AI training. These results highlight its potential to support increasingly large models efficiently. Moving forward, we will explore cost-effective training systems to further enhance LLM efficiency and scalability.

6 Limitations

Our BitDP system and fine-grained ultra-low bit quantization method has shown promising results on relatively large-scale models and datasets. However, due to computational resource constraints, the current experimental scale, while substantial, is still not sufficient to fully explore the method's potential—particularly on larger models, such as those with tens of billions of parameters, and more extensive datasets. Moreover, the additional overhead introduced by quantization and dequantization operations requires further optimization of computation operators to improve speed and efficiency.

While our work aims to reduce resource consumption for training large language models, we acknowledge the potential risk that by making training more resource-efficient, it could inadvertently encourage more widespread training of increasingly larger models, which might lead to greater cumulative environmental impact.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Wikimedia Foundation. Wikimedia downloads.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://github.com/jcpeterson/openwebtext`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA.

Conglong Li, Ammar Ahmad Awan, Hanlin Tang, Samyam Rajbhandari, and Yuxiong He. 2022. 1-bit lamb: communication efficient large-scale large-batch training with lamb's convergence speed. In *2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 272–281. IEEE.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*.

Yucheng Lu, Conglong Li, Minjia Zhang, Christopher De Sa, and Yuxiong He. 2023. Maximizing

communication efficiency for large-scale training via 0/1 adam. In *The Eleventh International Conference on Learning Representations*.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

NVIDIA Corporation. 2020. NVIDIA NCCL. https://developer.nvidia.com/nccl.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. 2023. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*.

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

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Shaohuai Shi, Xianhao Zhou, Shutao Song, Xingyao Wang, Zilin Zhu, Xue Huang, Xinan Jiang, Feihu Zhou, Zhenyu Guo, Liqiang Xie, et al. 2021. Towards scalable distributed training of deep learning on public cloud clusters. *Proceedings of Machine Learning and Systems*, 3:401–412.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 2021. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pages 10118–10129. PMLR.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Xiaoxia Wu, Connor Holmes, Zhewei Yao, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, et al. 2024. Zero++: Extremely efficient collective communication for large model training. In *The Twelfth International Conference on Learning Representations*.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. 2023. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Gradient Distribution Analysis of LLM Linear Layers

In this section, we analyze the gradient distributions of linear layers in large language models (LLMs) across different model sizes and layers. Specifically, we focus on two representative models: GPT-51M and GPT-13B, which differ significantly in scale and architecture. GPT-51M, a small-scale model, consists of 8 Transformer layers with a hidden dimension of 512. In contrast, GPT-13B, a large-scale model, contains 40 Transformer layers with a hidden dimension of 5120. The analysis spans 20,000 training steps, with gradient histograms sampled every 50 steps to capture the temporal evolution of gradient propagation. Figure 5 visualizes the overlaid histograms of weight gradients for all linear sub-layers within a single Transformer layer. The results are shown for GPT-51M at Layer 1 (first row) and GPT-13B at both Layer 1 (second row) and Layer 8 (third row). This figure provides insight into how gradient distributions evolve across models of different scales and depths.

From the analysis, we observe that the gradients exhibit a distinct triangular distribution, symmetrically centered around zero, across both small-scale and large-scale models. This distribution reflects stable gradient propagation throughout training, as gradients remain balanced without becoming excessively concentrated or dispersed. Notably, the triangular distributions are consistent across different model sizes and layers, as seen in GPT-51M at Layer 1 and GPT-13B at Layers 1 and 8. These findings suggest that larger models, such as GPT-13B, align more closely with our assumed gradient distribution. Such alignment makes them better suited for applying our system, further reinforcing the scalability of our approach.

B Proof of the Closed-form Threshold α^*

B.1 Problem Statement

For the 2-bit quantizer defined in Eq.(7) of the main text as

$$q_{i,j} = \begin{cases} 1, & g_{i,j} \geq \alpha_i, \\ 0, & -\alpha_i \leq g_{i,j} < \alpha_i, \\ -1, & g_{i,j} < -\alpha_i, \end{cases} \tag{11}$$

We determine the threshold α_i by solving the following optimization problem: maximizing

$$\mathcal{F}(\alpha_i) = \frac{1}{|\Omega_{\alpha_i}|}\left(\sum_{j\in\Omega_{\alpha_i}} |g_{i,j}|\right)^2, \tag{12}$$
$$\Omega_{\alpha_i} = \{\, j : |g_{i,j}| > \alpha_i \,\}.$$

This is under the condition that the gradient G follows a triangular distribution, where its probability density function (PDF) is

$$f_G(x) = \frac{h_i - |x|}{h_i^2}, \qquad x \in [-h_i, h_i], \tag{13}$$

where $h_i = \max_j |g_{i,j}|$. For $Y = |G|$,

$$f_Y(y) = \frac{2\,(h_i - y)}{h_i^2}, \qquad y \in [0, h_i]. \tag{14}$$

B.2 Continuous Form of the Objective

Replacing sums by expectations gives

$$\mathcal{F}(\alpha_i) \approx \frac{\left[\mathbb{E}(Y\mathbf{1}_{Y>\alpha_i})\right]^2}{\mathbb{P}(Y > \alpha_i)}. \tag{15}$$

Let $t = \alpha_i/h_i \in [0, 1)$. From Eq. (14),

$$\mathbb{P}(Y > \alpha_i) = \int_{\alpha_i}^{h_i} f_Y(y)\,dy = (1-t)^2, \tag{16}$$

$$\mathbb{E}\big[Y\mathbf{1}_{Y>\alpha_i}\big] = \int_{\alpha_i}^{h_i} y\, f_Y(y)\,dy = h_i\big(\tfrac{1}{3} - t^2 + \tfrac{2}{3}\,t^3\big). \tag{17}$$

Up to a factor of h_i^2, the objective function to maximize is

$$F(t) = \frac{\left[\frac{1}{3} - t^2 + \frac{2}{3}\,t^3\right]^2}{(1-t)^2} \tag{18}$$

Hence we maximize

$$f(t) = \frac{\frac{1}{3} - t^2 + \frac{2}{3}\,t^3}{1-t}, \qquad 0 \leq t < 1. \tag{19}$$

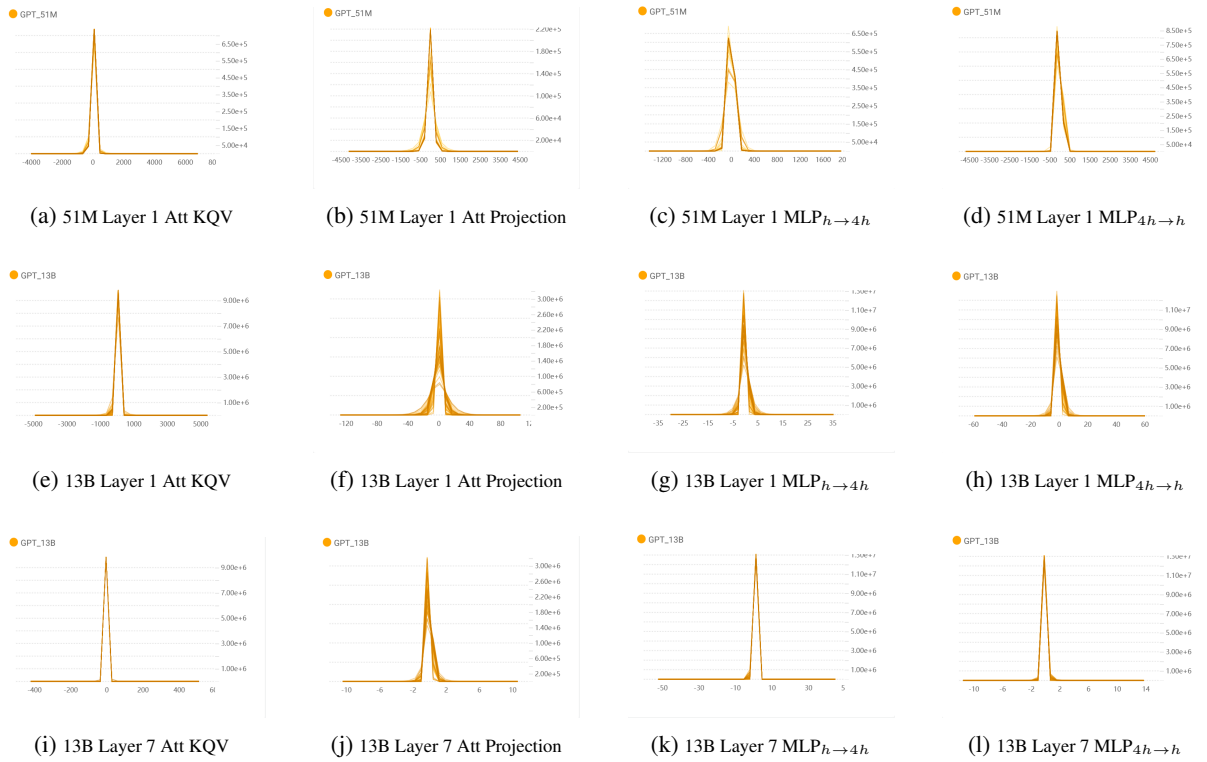
B.3 Solution

Write $n(t) = \frac{1}{3} - t^2 + \frac{2}{3}\,t^3$ and $d(t) = 1 - t$ so $f(t) = n(t)/d(t)$. Then

$$f'(t) = \frac{n'(t)(1-t) + n(t)}{(1-t)^2}, \tag{20}$$
$$n'(t) = 2t(t-1).$$

Setting $f'(t) = 0$ yields $4t^3 - 9t^2 + 6t - 1 = 0 = (t - \frac{1}{4})(4t^2 - 8t + 4)$. The only root in the interval $[0, 1)$ is $t^* = \frac{1}{4}$, giving the optimal threshold

$$\alpha_i^* = \frac{h_i}{4}. \tag{21}$$



(a) 51M Layer 1 Att KQV (b) 51M Layer 1 Att Projection (c) 51M Layer 1 MLP$_{h\to 4h}$ (d) 51M Layer 1 MLP$_{4h\to h}$

(e) 13B Layer 1 Att KQV (f) 13B Layer 1 Att Projection (g) 13B Layer 1 MLP$_{h\to 4h}$ (h) 13B Layer 1 MLP$_{4h\to h}$

(i) 13B Layer 7 Att KQV (j) 13B Layer 7 Att Projection (k) 13B Layer 7 MLP$_{h\to 4h}$ (l) 13B Layer 7 MLP$_{4h\to h}$

Figure 5: Overlaid gradient histograms of GPT-51M (51 million parameters) and GPT-13B (13 billion parameters) models across 20,000 training steps. Each plot overlays gradient distributions sampled every 50 steps during training, aggregated for different Transformer components (Attention KQV, Attention Projection, and MLP layers). The first row corresponds to GPT-51M at Layer 1, while the second and third rows represent GPT-13B at Layer 1 and Layer 8, respectively.

B.4 Relation to the Mean Absolute Gradient

For the triangular distribution, the mean of $Y = |G|$ is $\mathbb{E}[Y] = h_i/3$. Therefore,

$$\alpha_i^* = \frac{h_i}{4} = \frac{3}{4}\,\mathbb{E}_j[\,|g_{i,j}|\,]. \tag{22}$$

This relationship shows that α_i^* is $3/4$ of the mean absolute gradient, which offers greater robustness to outliers compared to an estimator based on the sample maximum h_i ($\alpha_i = (\max_j |g_{i,j}|)/4$). □

C Supplementary Information

We use three large-scale corpora for all experiments: BooksCorpus, English Wikipedia, and OpenWebText. BooksCorpus contains approximately 800M words of unpublished books, while for English Wikipedia, we only utilize the text passages, amounting to 2,500M words. These two datasets align with the training data used in BERT. Additionally, we incorporate OpenWebText, which consists of 38GB of text data (40GB using SI units) from 8,013,769 documents, matching the corpus used to train GPT-2. We sample from these datasets uniformly according to a ratio of 20% (BooksCorpus), 30% (Wikipedia), and 50% (OpenWebText). The combined corpus is split into training (94.9%), validation (5%), and test (0.1%) sets. This distribution provides a diverse and comprehensive foundation for evaluating our quantization methods across different linguistic contexts and domains.

All experiments with the above datasets are conducted using the PyTorch and Megatron-LM framework on NVIDIA V100 GPUs for research purposes, which is consistent with the intended use.

We utilized AI assistants for writing polishing.