LittleBit: Ultra Low-Bit Quantization via Latent Factorization

Banseok Lee* **Dongkyu Kim*** **Youngcheon You** **Youngmin Kim†**
Samsung Research
{bs93.lee, dongkyu.k, y01000.you, ym1012.kim}@samsung.com

Abstract

The deployment of large language models (LLMs) is frequently hindered by prohibitive memory and computational requirements. While quantization mitigates these bottlenecks, maintaining model fidelity in the sub-1-bit regime remains a persistent challenge. In this paper, we introduce LITTLEBIT, a novel framework for extreme LLM compression. We target quantization rates as low as 0.1 bits per weight (BPW), achieving a memory reduction of approximately $31\times$, which effectively compresses Llama2-13B to under 0.9 GB. We represent weights via low-rank latent matrix factorization and subsequently binarize the resulting factors. To counteract the information loss inherent to such drastic precision reduction, we integrate a multi-scale compensation mechanism that learns importance parameters across row, column, and latent dimensions. Two primary contributions enable effective training: Dual Sign-Value-Independent Decomposition (Dual-SVID) for quantization-aware training (QAT) initialization, and Residual Compensation to minimize approximation errors. Extensive experiments confirm the superiority of LITTLEBIT in the sub-1-bit domain; for instance, our method at 0.1 BPW surpasses the performance of leading techniques operating at 0.7 BPW on Llama2-7B. We establish a new size-performance trade-off—unlocking a potential $11.6\times$ inference speedup relative to FP16—and render powerful LLMs practical for resource-constrained environments. Our code is available at https://github.com/SamsungLabs/LittleBit.

1 Introduction

Large language models (LLMs) based on the Transformer architecture [1] have fundamentally transformed natural language processing. However, the scale of these models, often reaching hundreds of billions or trillions of parameters [2, 3], results in prohibitive computational and memory costs. In particular, the high demand for GPU VRAM hinders widespread deployment on consumer or edge devices [4–6].

Model quantization, which reduces numerical precision, serves as a primary technique to address these bottlenecks [7]. While post-training quantization (PTQ) methods such as GPTQ [8] and AWQ [9] effectively compress models to approximately 4 bits, achieving further compression (*e.g.*, to 1-bit) typically requires quantization-aware training (QAT) [10, 11] to maintain performance. QAT has demonstrated the capability to enable 1-bit compression while preserving model fidelity [12–14].

Nevertheless, even 1-bit models (*e.g.*, approximately 15.4 GB for 70B parameters [7]) may exceed the capacity of highly resource-constrained devices [15]. This limitation motivates the exploration of sub-1-bit quantization. Although prior work [16] has investigated this area, maintaining performance at extremely low effective bits (*e.g.*, 0.55 bits per weight (BPW)) while ensuring computational

*Equal contribution
†Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

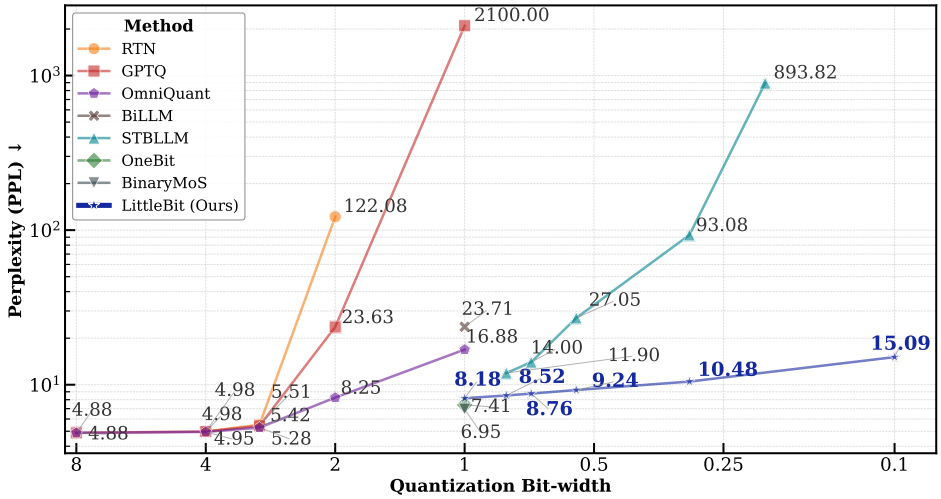


Figure 1: Low-bit quantization perplexity on Llama2-13B (WikiText-2). LITTLEBIT surpasses the state-of-the-art sub-1-bit quantization technique. Below 0.5 BPW, where the leading prior method degrades sharply, ours remains robust down to 0.1 BPW.

efficiency without restrictive hardware dependencies remains challenging [16]. Consequently, it is necessary to develop new techniques that achieve such extreme model compression, aiming for bit rates around 0.1 BPW, while preserving model performance.

We derive our approach from two observations. First, LLM weight matrices often exhibit pronounced low-rank structure [17, 18]. This observation suggests that factorization methods such as Singular Value Decomposition (SVD) [19] may offer a more stable compression pathway than pruning [20], particularly under extreme compression regimes. For instance, at compression ratios exceeding 50%, SVD-based approaches [21] consistently outperform pruning methods such as Wanda [22] (see Appendix E for details). Second, binarization inherently causes information loss [23]. Recent high-performing 1-bit methods [13, 24, 25] demonstrate that comprehensive scaling over multiple dimensions (*e.g.*, row and column) is crucial for mitigating this loss and stabilizing performance [25].

Building on these insights, we introduce **LITTLEBIT**, a novel method for extreme sub-1-bit quantization (*e.g.*, 0.1 BPW). In LITTLEBIT, we first represent weights via low-rank factorization ($\mathbf{W} \approx \mathbf{U}\mathbf{V}^\top$) and subsequently binarize these factors. To counteract errors from binarization, we leverage a multi-scale compensation mechanism that applies learnable scales across rows, columns, and an *additional latent* dimension. We complement this architecture with Residual Compensation.

Extensive experiments demonstrate the superiority of LITTLEBIT over STBLLM [16], the leading sub-1-bit technique, on LLMs ranging from 1.3B to 32B parameters. Notably, LITTLEBIT achieves competitive performance on Llama2-13B at 0.1 BPW (Fig. 1), surpassing STBLLM at 0.55 BPW. Furthermore, the 32B LITTLEBIT model at just 0.3 BPW maintains strong performance, substantially outperforming STBLLM and establishing a new state-of-the-art for sub-1-bit quantization.

In summary, our contributions are as follows:

- We propose LITTLEBIT, a novel framework unifying latent matrix factorization with multi-scale compensation to enable extreme sub-1-bit quantization. This method achieves effective bits down to 0.1 BPW while preserving model performance, establishing a viable size-performance trade-off for deploying LLMs in resource-constrained environments.
- We introduce Dual-SVID for the effective initialization of factorized structures and integrate Residual Compensation to mitigate approximation errors. These techniques collectively stabilize quantization-aware training (QAT) and counteract the information loss inherent in drastic precision reduction, ensuring robust learning dynamics.
- We provide extensive empirical validation demonstrating that LITTLEBIT consistently outperforms STBLLM [16], the prior leading sub-1-bit technique, across various model scales. Our results confirm that LITTLEBIT maintains superior fidelity in the deep compression regime, where baseline methods typically suffer severe degradation.

2 Related Works

2.1 Binarization and Quantization

Network binarization aims for extreme compression and acceleration by constraining weights or activations to ± 1. This enables efficient bitwise operations [26, 27]. While early studies such as BinaryConnect [28] and BNNs [26] demonstrated feasibility, they encountered substantial accuracy degradation [29]. Researchers partially mitigated this degradation by introducing scaling factors [30] and employing the Straight-Through Estimator (STE) [31] to handle the non-differentiable quantization function during training [32].

The direct application of these early techniques to large language models (LLMs) typically results in severe performance loss [33]. Consequently, LLM-specific quantization strategies have emerged. These are primarily categorized into post-training quantization (PTQ) and quantization-aware training (QAT) [34]. PTQ adapts pre-trained models and achieves competitive results at approximately 4-bit precision with methods such as GPTQ [8] and AWQ [9]. However, these methods generally struggle to maintain performance below 2-bit precision [33, 7]. The sub-1-bit regime presents particular challenges for PTQ methods due to severe information loss [35]. Recent work such as STBLLM [16] demonstrates that incorporating structured binarized compression can extend PTQ into the sub-1-bit regime, but a non-negligible accuracy gap persists.

In contrast, QAT integrates the quantization process directly into the training loop. This enables the model to learn and adapt to the low-precision format [36]. This approach generally yields superior performance compared to PTQ at very low bit-widths. Successful QAT approaches for low-bit scenarios often employ sophisticated scaling mechanisms alongside adaptive training, as evidenced by recent methods [13, 24]. Despite these advancements, consistently achieving high model fidelity when quantizing to less than 1.0 BPW remains a considerable challenge. This persisting difficulty underscores the need for novel architectures and training strategies specifically designed for such extreme compression regimes.

2.2 Low-Rank Approximation in Quantization

Beyond quantization, the inherent parameter redundancy within deep neural networks [35] can be effectively exploited using low-rank approximation methods. This redundancy is particularly evident in LLMs [37, 38]. Techniques such as Singular Value Decomposition (SVD) [19] allow large weight matrices to be represented as products of smaller factors. This offers an alternative compression strategy that may be more resilient than pruning, especially under high compression regimes [20]. Recognizing the potential for complementary benefits, researchers have explored combining low-rank factorization to reduce parameter count with quantization to lower parameter precision [39]. Initial studies applied SVD in various roles within the LLM context. Some approaches used activation statistics to guide decomposition [21], while others integrated SVD principles into initialization or parameter-efficient updates [40, 41, 13]. Further works applied transformations prior to factorization [42] or employed low-rank structures to model quantization errors [43]. A more integrated approach involves incorporating low-rank concepts directly within QAT methods. For example, DL-QAT [44] jointly learns a representation amenable to both low-rank approximation and low-precision quantization. However, this method is currently limited to 3-bit quantization. Optimizing structural properties such as rank and numerical precision during training represents a promising path toward effective LLM compression. There is potential for further exploration of even lower precision quantization.

3 Methodology

This section details **LittleBit**. It describes the factorized architecture with multi-scale compensation (Section 3.1), the Dual-SVID initialization for stable QAT (Section 3.2), and Residual Compensation for enhanced fidelity (Section 3.3). These components are optimized via QAT, as detailed in Section 4.

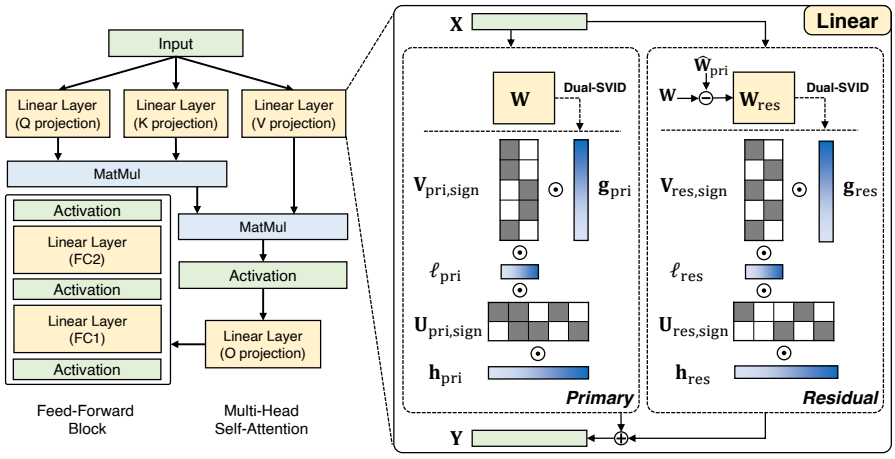


Figure 2: Comparison of a standard Transformer layer (left) and the LITTLEBIT architecture (right). LITTLEBIT performs linear transformation using parallel Primary and Residual pathways. The Primary path employs binarized factors ($\mathbf{U}_{\text{sign}}, \mathbf{V}_{\text{sign}}$) and FP16 scales ($\mathbf{h}, \mathbf{g}, \boldsymbol{\ell}$) on input $\mathbf{X}$, initialized from $\mathbf{W}$ via Dual-SVID. Simultaneously, the Residual path computes a correction with its own parameters ($\mathbf{U}_{\text{res,sign}}, \mathbf{V}_{\text{res,sign}}, \mathbf{h}_{\text{res}}, \mathbf{g}_{\text{res}}, \boldsymbol{\ell}_{\text{res}}$) from the approximation residual. Their outputs sum to form $\mathbf{Y}$, eliminating storage of the effective weight matrices $\widehat{\mathbf{W}}_{\text{pri}}$ and $\widehat{\mathbf{W}}_{\text{res}}$.

3.1 LITTLEBIT Architecture

The LITTLEBIT architecture, depicted in Fig. 2, redesigns linear layers for extreme compression (*e.g.*, 0.1 BPW) by synergistically combining low-rank factorization and binarization. The method leverages the observed low-rank structure in LLM weights [45] to approximate the weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ as:

$$\mathbf{W} \approx \mathbf{U}\mathbf{V}^\top \quad (\mathbf{U} \in \mathbb{R}^{d_{\text{out}} \times r}, \mathbf{V} \in \mathbb{R}^{d_{\text{in}} \times r}, r \ll \min(d_{\text{in}}, d_{\text{out}})) \tag{1}$$

We prioritize this factorization-based approach over other structural compression methods, such as pruning, due to its superior robustness in extreme compression regimes. As the comparative analysis against methods like Wanda demonstrates (Appendix E), SVD-based compression maintains model integrity far more effectively at high compression ratios. This provides a more stable foundation for ultra low-bit quantization. The resulting factors $\mathbf{U}$ and $\mathbf{V}$ are then binarized:

$$\mathbf{U}_{\text{sign}} = \text{sign}(\mathbf{U}), \quad \mathbf{V}_{\text{sign}} = \text{sign}(\mathbf{V}) \in \{-1, +1\} \tag{2}$$

To compensate for the significant information loss incurred by binarization [23], LITTLEBIT incorporates learnable FP16 scales. Beyond standard row ($\mathbf{h}$) and column ($\mathbf{g}$) scaling [33, 13, 25], the architecture introduces an *additional latent* scale ($\boldsymbol{\ell} \in \mathbb{R}^r$). This latent scale addresses the factorization structure by learning the relative importance of each of the r latent dimensions, which correspond to the columns of the binarized factors $\mathbf{U}_{\text{sign}}$ and $\mathbf{V}_{\text{sign}}$:

$$\mathbf{h} \in \mathbb{R}^{d_{\text{out}}}, \quad \mathbf{g} \in \mathbb{R}^{d_{\text{in}}}, \quad \boldsymbol{\ell} \in \mathbb{R}^r \tag{3}$$

The primary effective weight, $\widehat{\mathbf{W}}_{\text{pri}}$, is implicitly constructed from these components:

$$\widehat{\mathbf{W}}_{\text{pri}} = \text{diag}(\mathbf{h})\mathbf{U}_{\text{sign}}\text{diag}(\boldsymbol{\ell})\mathbf{V}_{\text{sign}}^\top\text{diag}(\mathbf{g}) \tag{4}$$

Instead of storing or optimizing $\widehat{\mathbf{W}}_{\text{pri}}$ directly, the model learns latent full-precision factors $\mathbf{U}$ and $\mathbf{V}$. These are binarized during the forward pass using Eq. (2), along with the FP16 scales $\mathbf{h}, \mathbf{g}, \boldsymbol{\ell}$. This factorized representation enables efficient computation during the forward pass.

Proposition 1 Let $\mathbf{X} \in \mathbb{R}^{\text{seq} \times d_{\text{in}}}$ be the input matrix. The forward computation $\mathbf{Y} = \mathbf{X}\widehat{\mathbf{W}}_{\text{pri}}^\top$ using the primary approximation (Eq. (4)) is efficiently computed as:

$$\mathbf{Y} = ((((\mathbf{X} \odot \mathbf{g})\mathbf{V}_{\text{sign}}) \odot \boldsymbol{\ell})\mathbf{U}_{\text{sign}}^\top) \odot \mathbf{h} \tag{5}$$

where $\odot$ denotes element-wise multiplication with broadcasting.

This decomposition (Eq. (5)) replaces a large high-precision General Matrix Multiply (GEMM) operation with two smaller binary matrix multiplications and element-wise scaling operations. This offers computational advantages and reduces memory requirements. The effective bits per weight (BPW) is determined by the storage cost of these learnable parameters relative to the original matrix size (see Appendix D for details).

3.2 Dual-SVID Initialization

Naively initializing the highly constrained LITTLEBIT structure can lead to unstable QAT. To circumvent this, we propose **Dual-SVID**, an SVD-based initialization method designed to provide a starting point where the initial effective weight $\widehat{\mathbf{W}}_{\text{pri},0}$ closely approximates $\mathbf{W}$. Dual-SVID aims to preserve essential sign and magnitude information from the optimal low-rank SVD factors (obtained from $\mathbf{W} \approx \mathbf{U}'\mathbf{V}'^{\top}$, then truncated) and map this information to the learnable LITTLEBIT parameters ($\mathbf{U}_{\text{sign}}, \mathbf{V}_{\text{sign}}, \mathbf{h}, \mathbf{g}, \boldsymbol{\ell}$). The process involves three steps: (1) Obtaining $\mathbf{U}', \mathbf{V}'$ via a truncated SVD. (2) Setting initial binary factors based on signs (Eq. (6)). (3) Decomposing the magnitudes $|\mathbf{U}'| \in \mathbb{R}^{d_{\text{out}} \times r}$ and $|\mathbf{V}'| \in \mathbb{R}^{d_{\text{in}} \times r}$. To initialize the scales, a rank-1 SVD (or equivalent rank-1 approximation) is performed on each of these magnitude matrices separately. For $|\mathbf{U}'|$, its best rank-1 approximation can be written as $\mathbf{s}_{U,0}(\boldsymbol{\ell}_{u,0})^{\top}$, where $\mathbf{s}_{U,0} \in \mathbb{R}^{d_{\text{out}}}$ becomes the initial row scale $\mathbf{h}_0$, and $\boldsymbol{\ell}_{u,0} \in \mathbb{R}^r$ is a component contributing to the latent scale. Similarly, for $|\mathbf{V}'|$, its rank-1 approximation $\mathbf{s}_{V,0}(\boldsymbol{\ell}_{v,0})^{\top}$ provides the initial column scale $\mathbf{g}_0 = \mathbf{s}_{V,0} \in \mathbb{R}^{d_{\text{in}}}$ and another latent scale component $\boldsymbol{\ell}_{v,0} \in \mathbb{R}^r$. These steps effectively disentangle multi-dimensional magnitude variations and allow for the estimation of initial row ($\mathbf{h}_0$), column ($\mathbf{g}_0$), and the final latent scale ($\boldsymbol{\ell}_0 = \boldsymbol{\ell}_{u,0} \odot \boldsymbol{\ell}_{v,0}$ by element-wise product), as shown in Eqs. (7) and (8):

$$\mathbf{U}_{\text{sign},0} = \text{sign}(\mathbf{U}'), \quad \mathbf{V}_{\text{sign},0} = \text{sign}(\mathbf{V}') \tag{6}$$

$$|\mathbf{U}'| \approx \mathbf{h}_0(\boldsymbol{\ell}_{u,0})^{\top}, \quad |\mathbf{V}'| \approx \mathbf{g}_0(\boldsymbol{\ell}_{v,0})^{\top} \tag{7}$$

$$\boldsymbol{\ell}_0 = \boldsymbol{\ell}_{u,0} \odot \boldsymbol{\ell}_{v,0} \tag{8}$$

The resulting initial effective weight $\widehat{\mathbf{W}}_{\text{pri},0} = \text{diag}(\mathbf{h}_0)\mathbf{U}_{\text{sign},0}\text{diag}(\boldsymbol{\ell}_0)\mathbf{V}_{\text{sign},0}^{\top}\text{diag}(\mathbf{g}_0)$ preserves key structural information. The learnable parameters are initialized as follows: the latent factors $\mathbf{U}$ and $\mathbf{V}$ are initialized with the factors $\mathbf{U}'$ and $\mathbf{V}'$ obtained from SVD, respectively, and the scales $\mathbf{h}, \mathbf{g}, \boldsymbol{\ell}$ are initialized to $\mathbf{h}_0, \mathbf{g}_0, \boldsymbol{\ell}_0$.

3.3 Residual Compensation

To further improve fidelity without increasing the primary path's rank, we introduce **Residual Compensation**. This technique is motivated by the theoretical insight that a two-stage error correction can be more effective than a single, larger approximation (Appendix A). Crucially, this method does not increase the model's total parameter budget; instead, the fixed bit budget of a single, higher-rank approximation is strategically reallocated into two lower-rank paths: a primary and a residual path. The term 'residual' primarily describes its role during initialization, where the auxiliary path is configured to model the error of the primary approximation. During QAT, both paths are optimized jointly to collectively represent the original weight. The advantage of this dual-path approach over a single path is empirically validated in our ablation studies (Appendix B.1).

To implement this, Residual Compensation employs a parallel auxiliary path that learns an approximation $\widehat{\mathbf{W}}_{\text{res}}$ of the residual error. This auxiliary path mirrors the structure of the LITTLEBIT path:

$$\widehat{\mathbf{W}}_{\text{res}} = \text{diag}(\mathbf{h}_{\text{res}})\mathbf{U}_{\text{res,sign}}\text{diag}(\boldsymbol{\ell}_{\text{res}})\mathbf{V}_{\text{res,sign}}^{\top}\text{diag}(\mathbf{g}_{\text{res}}) \tag{9}$$

The parameters of this auxiliary path ($\mathbf{U}_{\text{res,sign}}, \mathbf{V}_{\text{res,sign}}, \mathbf{h}_{\text{res}}, \mathbf{g}_{\text{res}}, \boldsymbol{\ell}_{\text{res}}$) are also learnable during QAT. They are initialized using the Dual-SVID strategy applied to the initial residual error $\mathbf{W}_{\text{res},0} = \mathbf{W} - \widehat{\mathbf{W}}_{\text{pri},0}$. Employing a separate path allows the model to specifically learn and compensate for the potentially distinct characteristics of this residual error. The final effective weight is the sum of both approximations:

$$\widehat{\mathbf{W}} = \widehat{\mathbf{W}}_{\text{pri}} + \widehat{\mathbf{W}}_{\text{res}} \tag{10}$$

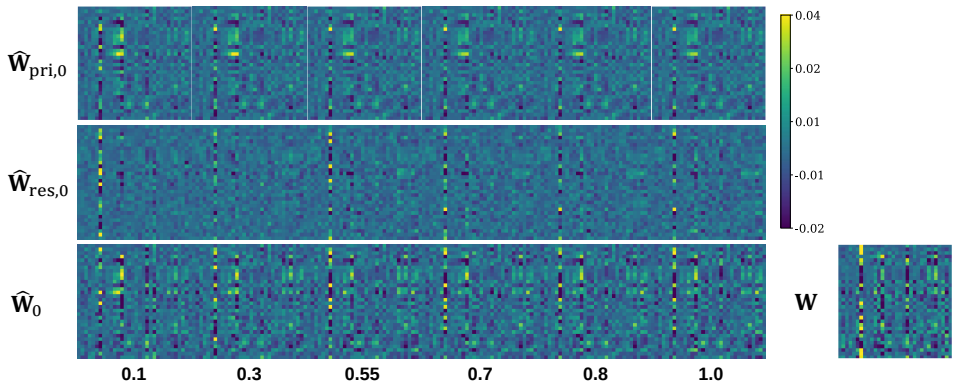


Figure 3: Visualization of Dual-SVID initialized weight components for a selected layer in Llama2-7B (Query Weight, Layer 0). Columns, from left to right, represent effective bits of 0.1, 0.3, 0.55, 0.7, 0.8, and 1.0 BPW. Rows display the primary approximation ($\widehat{\mathbf{W}}_{\text{pri},0}$), the residual approximation ($\widehat{\mathbf{W}}_{\text{res},0}$), and their sum ($\widehat{\mathbf{W}}_0 = \widehat{\mathbf{W}}_{\text{pri},0} + \widehat{\mathbf{W}}_{\text{res},0}$). The rightmost image shows the corresponding crop of the original weight matrix ($\mathbf{W}$) for reference.

This targeted error correction is often advantageous for maintaining performance at extreme compression levels, as demonstrated by the ablation studies presented in Appendix B.1. It is important to note that all learnable parameters from both the primary and residual paths are included when calculating the final BPW (see Appendix D for calculation details).

Figure 3 visualizes the initial state of weight components ($\widehat{\mathbf{W}}_{\text{pri},0}$, $\widehat{\mathbf{W}}_{\text{res},0}$, and their sum $\widehat{\mathbf{W}}_0$) for a Llama2-7B query weight, derived via Dual-SVID across effective bits from 0.1 to 1.0, compared against the original weight $\mathbf{W}$ (far right). Even at a low 0.1 BPW, the primary approximation $\widehat{\mathbf{W}}_{\text{pri},0}$ (top row, leftmost) captures the dominant low-rank structure of $\mathbf{W}$, albeit with considerable simplification. As effective bits increase, $\widehat{\mathbf{W}}_{\text{pri},0}$ progressively refines details and more effectively suppresses disruptive approximation noise, allowing underlying weight patterns to emerge with greater clarity. A key observation is that $\widehat{\mathbf{W}}_0$ (bottom row) at a modest 0.3 BPW—benefiting from the initialized residual component $\widehat{\mathbf{W}}_{\text{res},0}$ (middle row)—often presents a more faithful initial representation of $\mathbf{W}$ than $\widehat{\mathbf{W}}_{\text{pri},0}$ alone even at a higher 1.0 BPW. This underscores that Residual Compensation, even at the initialization stage, is vital for capturing complexities missed by the primary low-rank approximation, thereby providing a more faithful starting point for QAT.

4 Experiments

4.1 Settings

Evaluation Setup We evaluated LittleBit across diverse LLM families, including Llama [46], Llama2 [47], Llama3 [48], OPT [49], Phi-4 [50], and QwQ [51]. These models span parameter scales from 1.3B to 32B. Perplexity (PPL) on the WikiText-2 [52] validation dataset served as the primary performance metric. Appendix C provides additional results on the C4 [53] and PTB [54] datasets. Furthermore, we assessed zero-shot accuracy on common reasoning benchmarks: WinoGrande [55], OpenBookQA (OBQA) [56], HellaSwag [57], BoolQ [58], ARC-Easy (ARC-e), ARC-Challenge (ARC-c) [59], and PIQA [60].

Training Details We optimized the LittleBit model parameters, initialized via the Dual-SVID method (Section 3.2), using QAT with knowledge distillation (KD) [61, 36, 62]. The original pre-trained full-precision model functioned as the teacher ($\mathcal{T}$) for the LittleBit student model ($\mathcal{S}$). The QAT objective combines the standard output Kullback-Leibler (KL) divergence loss, $\mathcal{L}_{\text{out}}$, and an intermediate layer mean squared error (MSE) loss, $\mathcal{L}_{\text{inter}}$, to match hidden representations. We weighted these terms using an empirically determined coefficient $\lambda = 10$:

$$\mathcal{L}_{\text{QAT}} = \mathcal{L}_{\text{out}} + \lambda \mathcal{L}_{\text{inter}}. \tag{11}$$

Table 1: Perplexity (PPL) comparison on WikiText-2 across various LLMs and quantization methods. Lower PPL indicates better performance. BPW denotes effective bits per weight. Methods marked with † use quantization-aware training (QAT). STBLLM [16] utilizes N:M sparsity (ratio in parentheses). LITTLEBIT demonstrates strong performance, particularly in the sub-1-bit regime.

Settings			OPT	Llama		Llama2		Llama3	Phi-4	QwQ
Method	Block Size	BPW	1.3B	7B	13B	7B	13B	8B	14.7B	32B
FullPrecision	-	16	14.62	5.67	5.09	5.47	4.88	6.10	6.67	6.34
OmniQuant	-	2	28.82	9.75	7.83	11.20	8.25	349.27	12.09	10.19
GPTQ	128	2	119.98	39.96	15.08	52.22	23.63	1.5e3	24.96	67.16
BiLLM	128	1.1	74.07	41.95	13.95	29.00	23.71	54.29	16.95	15.4
OneBit†	-	1	20.27	8.21	7.37	8.36	7.41	13.09	9.92	9.86
BinaryMoS†	-	1	18.09	7.84	7.05	7.74	6.95	10.83	9.51	8.99
LITTLEBIT †	-	1	20.33	9.03	8.17	9.08	8.18	16.30	11.28	12.08
STBLLM	128	0.80 (6:8)	52.13	15.19	9.43	13.81	11.90	30.90	12.12	12.19
STBLLM	128	0.70 (5:8)	73.42	19.52	11.47	19.17	14.00	59.83	14.57	13.78
STBLLM	128	0.55 (4:8)	123.03	38.73	16.88	30.67	27.05	241.95	21.99	18.32
STBLLM	128	0.30 (2:8)	2.3e3	1.6e3	592.06	1.8e3	893.82	1.7e5	761.05	512.01
LITTLEBIT †	-	0.80	21.32	9.44	8.50	9.53	8.52	17.28	11.70	12.46
LITTLEBIT †	-	0.70	21.99	9.66	8.71	9.85	8.76	18.01	12.05	13.22
LITTLEBIT †	-	0.55	23.35	10.09	9.16	10.47	9.24	18.47	12.80	13.57
LITTLEBIT †	-	0.30	28.30	11.50	10.33	12.00	10.48	20.34	14.71	16.48
LITTLEBIT †	-	0.10	53.76	15.58	13.71	15.92	15.09	26.11	19.73	35.26

Adhering to the protocol in [24], the training data combined WikiText-2 with selected partitions from C4. The configuration included a sequence length of 2048 tokens, 5 epochs, the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) [63], and a cosine learning rate decay with 2% warm-up (see Appendix G for details). For models employing Grouped-Query Attention (GQA) [64], such as Llama3, Phi-4, and QwQ, we specifically adjusted the latent ranks r for key (K) and value (V) projection layers (see Appendix B.3) to maintain performance under extreme quantization. We employed *SmoothSign* (forward: $\text{sign}(x)$; backward gradient: $\tanh(kx)$, $k = 100$) for backpropagation, as it demonstrated superior performance compared to the Straight-Through Estimator (STE) [31] (see Appendix B.2). Although proxy gradients are established in the literature, this work validates the specific application of the $\tanh(100x)$ derivative as a stable gradient for QAT in the ultra-low-bit regime. Our ablation study confirms the effectiveness of this approach (Appendix B.2).

Baselines We benchmarked LITTLEBIT across effective bit rates ranging from approximately 1.0 down to 0.1 bits per weight (BPW). In the challenging sub-1-bit regime, we compared our method against STBLLM [16], a post-training quantization (PTQ) method that employs N:M sparsity. For settings approximating 1.0 BPW, we compared against the 1-bit PTQ method BiLLM [7] and quantization-aware training (QAT) approaches such as OneBit [13] and BinaryMoS [24]. We included results from standard low-bit methods, such as GPTQ [8] and OmniQuant [65], to provide broader context.

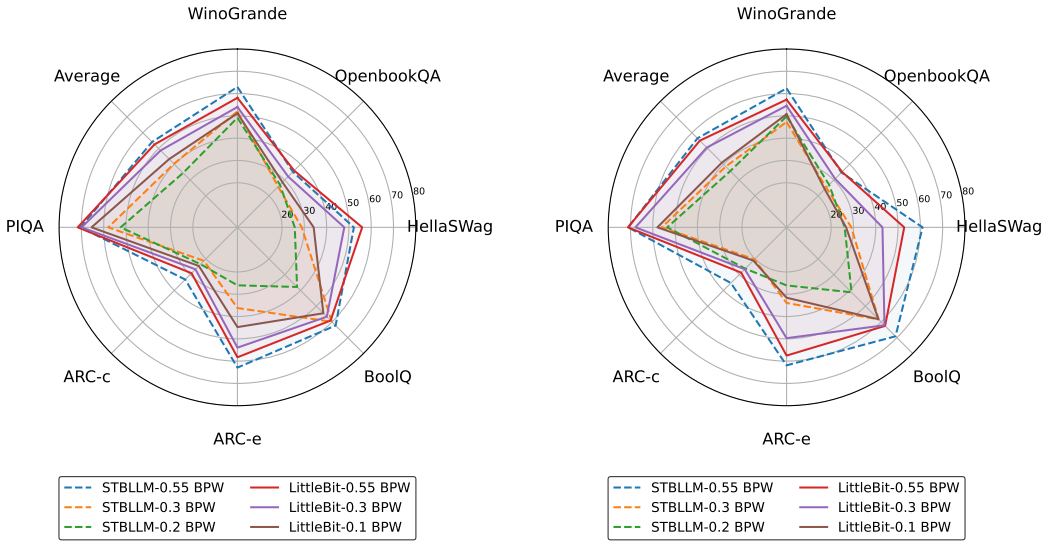
4.2 Main Results

Superior Perplexity in Sub-1-bit Regime Table 1 presents the perplexity (PPL) results of LITTLEBIT compared to baseline methods. The proposed method demonstrates robust performance, particularly within the sub-1-bit regime. Relative to STBLLM [16], LITTLEBIT achieves markedly superior PPL scores at 0.8, 0.7, and 0.55 BPW. For instance, on Llama2-7B, LITTLEBIT at 0.55 BPW achieves a PPL of 10.47, which represents a substantial improvement over the 30.67 score reported for STBLLM. We observe comparable gains across other models and bit-widths. For Llama2-13B at 0.8 BPW, LITTLEBIT records a PPL of 8.52, whereas STBLLM records 11.90.

Extreme Low-BPW Stability A primary advantage of LITTLEBIT lies in its exceptional stability and performance in the extreme sub-0.5 BPW range, specifically at 0.3 BPW and 0.1 BPW. In this regime, methods such as STBLLM exhibit severe performance degradation. For example, Llama2-7B yields a PPL of 1.8×10^3 at 0.3 BPW. Conversely, LITTLEBIT maintains strong PPL scores, such as 12.00 for Llama2-7B at 0.3 BPW and 15.92 at an unprecedented rate of 0.1 BPW. When configured for effective bits approximating 1.0 BPW by adjusting the latent rank r, LITTLEBIT yields perplexity

Table 2: Zero-shot accuracy (%) comparison on common sense reasoning benchmarks. Compares Full Precision (FP16) models against STBLLM [16] and LittleBit at 0.55 and 0.3 BPW.

Models	Method	WinoGrande	OBQA	HellaSwag	BoolQ	ARC-e	ARC-c	PIQA	Average
Llama2-7B	FullPrecision	67.16	40.80	72.95	71.37	69.44	40.95	78.12	62.97
	STBLLM (0.55)	52.80	33.00	36.94	62.17	37.33	25.34	62.46	44.29
	STBLLM (0.3)	50.43	31.80	26.20	37.83	25.88	25.09	53.26	35.78
	LittleBit (0.55)	51.62	34.00	44.57	61.38	44.15	26.96	68.12	**47.26**
	LittleBit (0.3)	51.30	34.80	37.61	61.80	39.98	25.09	65.83	**45.20**
Llama2-13B	FullPrecision	69.45	41.80	76.58	69.29	73.19	44.53	78.61	64.78
	STBLLM (0.55)	55.25	31.20	35.10	62.23	42.51	27.22	61.75	45.04
	STBLLM (0.3)	51.54	28.80	26.01	55.63	27.53	24.74	53.32	38.22
	LittleBit (0.55)	53.04	35.60	51.06	50.58	46.09	29.10	70.78	**48.03**
	LittleBit (0.3)	51.30	34.00	43.81	55.96	42.59	25.17	68.77	**45.94**
Llama3-8B	FullPrecision	72.92	45.00	79.18	81.25	80.21	52.98	79.54	70.15
	STBLLM (0.55)	52.17	25.80	30.61	57.16	30.35	23.72	57.56	39.62
	STBLLM (0.3)	49.57	26.60	26.36	51.62	26.05	24.40	51.74	36.62
	LittleBit (0.55)	50.12	30.20	36.78	57.55	46.80	22.95	66.27	**44.38**
	LittleBit (0.3)	51.93	28.20	33.91	57.98	43.48	24.32	64.64	**43.49**



(a) Phi-4 14B (b) QwQ-32B

Figure 4: Zero-shot accuracy (%) on 7 common sense reasoning tasks. The comparison highlights the robustness of LittleBit (solid lines) against STBLLM (dashed lines) in the sub-0.5 BPW regime. While STBLLM degrades sharply, LittleBit preserves viable reasoning capabilities for both (a) Phi-4 and (b) QwQ models.

scores comparable to specialized 1-bit QAT methods. For example, on Llama2-7B, LittleBit achieves 9.08 compared to 8.36 for OneBit [13]. It also remains competitive with PTQ methods such as BiLLM [7]. Although BinaryMoS [24] may exhibit slightly lower PPL in certain 1-bit scenarios (*e.g.*, Llama2-7B: BinaryMoS 7.74), this performance difference is attributable to its use of more complex mechanisms, such as dynamic scaling, which represent a distinct set of architectural and computational trade-offs. Our analysis of the trade-off involved in adjusting the latent rank to tune effective bits from 1.0 down to 0.1 BPW indicates that performance degradation is minimal down to approximately 0.55 BPW. However, a quantization cliff appears between 0.3 and 0.1 BPW. This observation positions the 0.3–0.55 BPW range as an optimal sweet spot for balancing compression and accuracy.

Model Generalization & Zero-Shot We consistently observe the advantages of LittleBit across a diverse set of model architectures and sizes. This includes the Llama3 family, which is typically considered challenging for quantization, as well as other contemporary models such as Phi-4. For example, Llama3-8B quantized by LittleBit to 0.55 BPW yields a PPL of 18.47. This result substantially outperforms the STBLLM score of 241.95 at 0.55 BPW with 4:8 sparsity. The zero-shot

Table 3: Memory footprint comparison (GB) for Llama2 [47] models under different quantization methods. LITTLEBIT is evaluated at various BPWs, achieved by adjusting the latent rank r. Compression factors relative to FP16 are shown in parentheses.The gap between the effective BPW and the compression factor arises because quantization is applied only to the Transformer blocks, while components like the embedding layer and the lm_head remain in FP16.

Model	FP16	BiLLM [7]	OneBit [13]	LITTLEBIT (Ours)			
BPW	16	1.1	1	0.8	0.55	0.3	0.1
Llama2-7B	13.49 GB	1.60 GB (8.43×)	1.36 GB (9.92×)	1.19 GB (11.34×)	0.98 GB (13.77×)	0.79 GB (17.08×)	0.63 GB (21.41×)
Llama2-13B	26.06 GB	2.80 GB (9.31×)	2.28 GB (11.43×)	1.95 GB (13.36×)	1.51 GB (17.26×)	1.15 GB (22.66×)	0.84 GB (31.02×)
Llama2-70B	138.04 GB	15.40 GB (8.96×)	9.72 GB (14.20×)	7.97 GB (17.31×)	5.83 GB (23.68×)	3.70 GB (37.31×)	1.98 GB (69.72×)

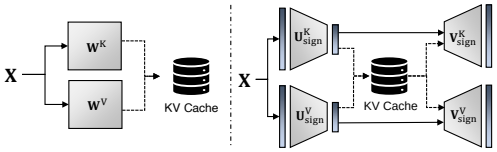


Figure 5: Conceptual view of KV Cache storage: the standard method (left) stores the full hidden dimension (d_{model}), whereas LITTLEBIT (right) caches a reduced latent dimension (r)

Table 4: Estimated KV cache memory reduction for Llama2-7B using LITTLEBIT, with a reduction factor of approximately d_{model}/r.

BPW	KV Latent Rank (r)	KV Cache Reduction Factor
0.80	1,624	≈2.5×
0.55	1,112	≈3.7×
0.30	600	≈6.8×
0.10	192	≈21.3×

evaluation results presented in Table 2 corroborate these perplexity-based findings. For instance, Llama2-7B compressed by LITTLEBIT to 0.55 BPW achieves a mean accuracy of 47.26%, and the model maintains 45.20% mean accuracy even at an extreme 0.3 BPW. These results compare favorably with those of STBLLM at similar or even higher effective bits. STBLLM on Llama2-7B at 0.55 BPW scores 44.29%. This outcome suggests superior preservation of the intrinsic reasoning capabilities of the models even under extreme compression. It highlights the efficacy of LITTLEBIT in creating highly compact yet powerful LLMs.

4.3 Reasoning Performance of Extremely Compressed LLMs

The capability of LITTLEBIT to achieve sub-1-bit compression while maintaining robust model fidelity prompts an evaluation of its reasoning abilities, particularly for large-scale models under extreme compression. We investigated the Phi-4 14B model and extended our analysis to the QwQ-32B model, comparing their performance against STBLLM across 7 zero-shot reasoning tasks.

When compressed with LITTLEBIT to 0.55 BPW, Phi-4 achieved an average accuracy of approximately 52.3%. Although this performance is marginally lower than STBLLM at a similar compression level (54.2% at 0.55 BPW), LITTLEBIT demonstrates substantially more graceful degradation. At the more aggressive 0.3 BPW setting, the LITTLEBIT-compressed Phi-4 maintained an average accuracy of 48.7%, which markedly outperforms STBLLM at 0.3 BPW (40.3%). Even at an extreme 0.1 BPW, Phi-4 retained a notable accuracy of 43.6%. We observed a consistent trend with the QwQ-32B model, as illustrated in Figure 4. While STBLLM exhibited severe performance collapse in the sub-0.5 BPW regime, LITTLEBIT maintained a stable trajectory down to 0.1 BPW. These results highlight the ability of LITTLEBIT to preserve robust reasoning capabilities across varying model scales as compression becomes increasingly extreme.

5 Analysis

Memory Footprint Reduction LITTLEBIT is designed for resource-constrained environments, such as on-device deployment, and achieves substantial reductions in model memory footprint. As summarized in Table 3, quantizing a Llama2-7B model (originally 13.49 GB in FP16) to an effective bit rate of 0.3 BPW using LITTLEBIT reduces its required storage to 0.79 GB. This corresponds to a compression factor exceeding $17\times$. At the extreme setting of 0.1 BPW, the footprint is further reduced to 0.63 GB (over $21\times$ compression). We observe comparable reductions for larger models, such as Llama2-70B, where the memory footprint decreases from 138.04 GB to under 2 GB at 0.1 BPW, achieving nearly $70\times$ compression. These substantial memory reductions significantly expand the range of devices capable of hosting large-scale language models, thereby enabling complex language tasks on hardware with limited VRAM or storage capacity.

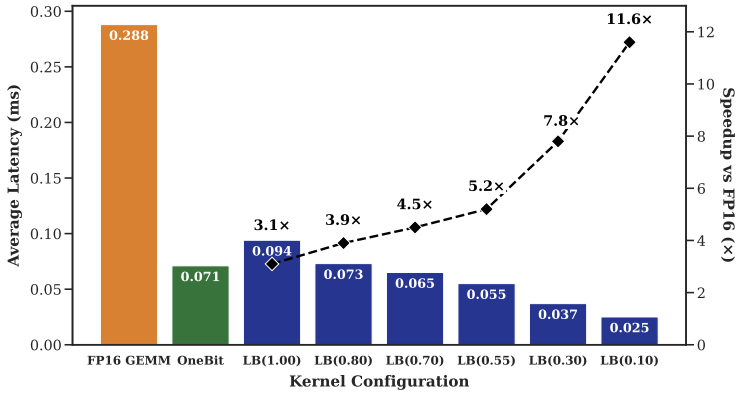


Figure 6: Kernel-level latency (bars, left axis) and speedup relative to FP16 (dashed line, right axis) on an NVIDIA A100 GPU for a Llama2-70B MLP layer ($8,192 \times 28,672$). The benchmark compares the FP16 GEMM baseline with OneBit and the proposed LITTLEBIT (LB) kernel across various BPW configurations ranging from 1.0 to 0.1.

KV Cache Compression The proposed factorization (Eq. (4)) inherently compresses the KV cache when applied to key and value projection matrices. Because the forward computation (Eq. (5)) operates on latent states, intermediate rank r vectors are cached rather than full d_{model}-dimensional vectors, as illustrated in Fig. 5. Consequently, KV cache memory requirements are reduced by a factor of approximately d_{model}/r. For example, a reduction of up to $21.3\times$ is observed for Llama2-7B at 0.1 BPW (Table 4). While this effect aligns with explicit compression methods such as MLA [66] and ASVD [42], the approach described herein achieves this via unified factorization within attention layers, thereby reducing both weight and activation memory simultaneously.

Latency Considerations In addition to memory optimization, inference latency presents a critical performance metric. The factorized architecture (Eq. (5)) facilitates computational acceleration through low-rank binary operations, distinct from methods solely prioritizing memory or requiring specific hardware. To evaluate this capability, a custom 1-bit GEMV CUDA kernel was developed to accelerate the primary computational path. As illustrated in Fig. 6 for a Llama2-70B MLP layer, the kernel demonstrates speedups that inversely correlate with bit-width, peaking at **11.6**$\times$ over an optimized FP16 baseline at 0.1 BPW.

These findings suggest the proposed method enables efficient deployment with reduced latency. The discrepancy between theoretical computational gains and practical latency is likely attributable to memory access dominance during small-batch inference. The factorized approach involves multiple memory-bound operations, and the custom kernel has not yet reached the optimization level of industry-standard libraries. Appendix H provides a comprehensive analysis covering theoretical FLOPs, practical latency, and end-to-end decoding throughput.

6 Conclusion

This study proposed LITTLEBIT, a method designed to advance LLM compression into the sub-0.5 BPW regime while maintaining viable performance at 0.1 BPW. Performance improvements were realized through SVD-inspired latent matrix factorization, factor binarization, and a multi-scale compensation mechanism encompassing row, column, and latent dimensions. Furthermore, Dual-SVID initialization was introduced to stabilize quantization-aware training (QAT), alongside Residual Compensation for error mitigation. Empirical results demonstrate that the proposed method markedly outperforms prior sub-1-bit techniques and preserves fidelity across LLMs scaling up to 32B parameters. Consequently, LITTLEBIT suggests a favorable size–performance trade-off, thereby facilitating LLM deployment in resource-constrained environments. Future research may address practical deployment through hardware co-design for edge devices, such as neural processing units (NPUs). Additional improvements in model fidelity could be examined by integrating advanced techniques such as non-uniform bit allocation, applied either across model layers or within single layers via dynamic bit budget distribution between primary and residual paths.

Acknowledgements

We are grateful to Dr. Heonjae Ha, Dr. SangJeong Lee, Minseop Choi, Changdong Kim, and Hyochan Chong for their insightful discussions and generous support in this work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017.

[2] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 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, 2021.

[3] Mikhail Isaev, Nic McDonald, and Richard Vuduc. Scaling Infrastructure to Support Multi-Trillion Parameter LLM Training. In *Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023)*, 2023.

[4] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A Review on Edge Large Language Models: Design, Execution, and Applications. *ACM Computing Surveys*, 57(8):1–35, 2025.

[5] Xiurui Pan, Endian Li, Qiao Li, Shengwen Liang, Yizhou Shan, Ke Zhou, Yingwei Luo, Xiaolin Wang, and Jie Zhang. InstInfer: In-Storage Attention Offloading for Cost-Effective Long-Context LLM Inference. *arXiv preprint arXiv:2409.04992*, 2024.

[6] Libo Zhang, Zhaoning Zhang, Rui Li, Zhiliang Tian, Songzhu Mei, Dongsheng Li, et al. Dovetail: A CPU/GPU Heterogeneous Speculative Decoding for LLM inference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17393–17406, 2025.

[7] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. BiLLM: Pushing the Limit of Post-Training Quantization for LLMs. *arXiv preprint arXiv:2402.04291*, 2024.

[8] 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.

[9] 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 On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[10] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[11] Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. QKD: Quantization-aware Knowledge Distillation. *arXiv preprint arXiv:1911.12491*, 2019.

[12] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit Transformers for Large Language Models. *arXiv preprint arXiv:2310.11453*, 2023.

[13] Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. OneBit: Towards Extremely Low-bit Large Language Models. *Advances in Neural Information Processing Systems*, 37:66357–66382, 2024.

[14] Liqun Ma, Mingjie Sun, and Zhiqiang Shen. FBI-LLM: Scaling up Fully Binarized LLMs from Scratch via Autoregressive Distillation. *arXiv preprint arXiv:2407.07093*, 2024.

[15] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A Review on Edge Large Language Models: Design, Execution, and Applications. *ACM Computing Surveys*, 57(8):1–35, 2025.

[16] Peijie Dong, Lujun Li, Yuedong Zhong, Dayou Du, Ruibo Fan, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Yike Guo, et al. STBLLM: Breaking the 1-bit Barrier with Structured Binary LLMs. *arXiv preprint arXiv:2408.01803*, 2024.

[17] Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. From Low Rank Gradient Subspace Stabilization to Low-Rank Weights: Observations, Theories, and Applications. *arXiv preprint arXiv:2407.11239*, 2024.

[18] Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Yiran Chen, Kurt Keutzer, and Chenfeng Xu. Dobi-SVD: Differentiable SVD for LLM Compression and Some New Perspectives. *arXiv preprint arXiv:2502.02723*, 2025.

[19] Gene H Golub and Christian Reinsch. Singular Value Decomposition and Least Squares Solutions. In *Linear algebra*, pages 134–151. Springer, 1971.

[20] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149*, 2015.

[21] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: Optimizing Singular Value Truncation for Large Language Model Compression. *arXiv preprint arXiv:2503.12340*, 2025.

[22] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models. *arXiv preprint arXiv:2306.11695*, 2023.

[23] Ruihao Gong, Yifu Ding, Zining Wang, Chengtao Lv, Xingyu Zheng, Jinyang Du, Yang Yong, Shiqiao Gu, Haotong Qin, Jinyang Guo, et al. A Survey of Low-bit Large Language Models: Basics, Systems, and Algorithms. *Neural Networks*, page 107856, 2025.

[24] Dongwon Jo, Taesu Kim, Yulhwa Kim, and Jae-Joon Kim. Mixture of Scales: Memory-Efficient Token-Adaptive Binarization for Large Language Models. *Advances in Neural Information Processing System*, 37:137474–137494, 2024.

[25] Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Linghe Kong, Yulun Zhang, Xiaokang Yang, et al. ARB-LLM: Alternating Refined Binarizations for Large Language Models. *arXiv preprint arXiv:2410.03129*, 2024.

[26] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[27] Cheng Fu, Shilin Zhu, Hao Su, Ching-En Lee, and Jishen Zhao. Towards Fast and Energy-Efficient Binarized Neural Network Inference on FPGA. *arXiv preprint arXiv:1810.02068*, 2018.

[28] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations. volume 28, 2015.

[29] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary Neural Networks: A Survey. *Pattern Recognition*, 105:107281, 2020.

[30] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[31] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.

[32] Matt Schoenbauer, Daniele Moro, Lukasz Lew, and Andrew Howard. Custom Gradient Estimators are Straight-Through Estimators in Disguise. *arXiv preprint arXiv:2405.05171*, 2024.

[33] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. PB-LLM: Partially Binarized Large Language Models. *arXiv preprint arXiv:2310.00034*, 2023.

[34] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. In *Low-power computer vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[35] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting Parameters in Deep Learning. *Advances in Neural Information Processing Systems*, 26, 2013.

[36] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484, 2024.

[37] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, 2025.

[38] Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What Matters in Transformers? Not All Attention is Needed. *arXiv preprint arXiv:2406.15786*, 2024.

[39] Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing Large Language Models using Low Rank and Low Precision Decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

[40] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[41] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

[42] Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware Singular Value Decomposition for Compressing Large Language Models. *arXiv preprint arXiv:2312.05821*, 2023.

[43] Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. LQER: Low-Rank Quantization Error Reconstruction for LLMs. *arXiv preprint arXiv:2402.02446*, 2024.

[44] Wenjing Ke, Zhe Li, Dong Li, Lu Tian, and Emad Barsoum. DL-QAT: Weight-Decomposed Low-Rank Quantization-Aware Training for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 113–119, 2024.

[45] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware Singular Value Decomposition for Large Language Model Compression. *arXiv preprint arXiv:2403.07378*, 2024.

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

[47] 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.

[48] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

[49] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*, 2022.

[50] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 Technical Report. *arXiv preprint arXiv:2412.08905*, 2024.

[51] Qwen Team. QwQ-32B: Embracing the Power of Reinforcement Learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

[52] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. *arXiv preprint arXiv:1609.07843*, 2016.

[53] 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. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[54] Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, page 114–119. Association for Computational Linguistics, 1994.

[55] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Communications of the ACM*, 64(9):99–106, 2021.

[56] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *arXiv preprint arXiv:1809.02789*, 2018.

[57] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[58] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *arXiv preprint arXiv:1905.10044*, 2019.

[59] 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. *arXiv preprint arXiv:1803.05457*, 2018.

[60] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

[62] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. TernaryBERT: Distillation-aware Ultra-low Bit BERT. *arXiv preprint arXiv:2009.12812*, 2020.

[63] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*, 2015.

[64] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[65] 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. *arXiv preprint arXiv:2308.13137*, 2023.

[66] DeepSeek-AI. DeepSeek-V2: A Strong Mixture-of-Experts Language Model with Expert Attention. *arXiv preprint arXiv:2405.04434*, 2024.

[67] Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv:1911.02150*, 2019.

A Mathematical Analysis and Proofs

This section provides mathematical details for the LITTLEBIT computation and explores theoretical aspects related to low-rank quantization errors, offering motivation for certain design choices.

Proposition 1 (LITTLEBIT Forward Pass Computation (Proof Detail))**.** *As stated in Section 3.1 (Proposition 1), for input* $\mathbf{X} \in \mathbb{R}^{\text{seq}\times d_{\text{in}}}$*, the primary quantized weight matrix is* $\widehat{\mathbf{W}}_{\text{pri}} = \text{diag}(\mathbf{h})\mathbf{U}_{\text{sign}}\text{diag}(\boldsymbol{\ell})\mathbf{V}_{\text{sign}}^{\top}\text{diag}(\mathbf{g})$*. The forward pass* $\mathbf{Y} = \mathbf{X}\widehat{\mathbf{W}}_{\text{pri}}^{\top}$ *can be computed as shown in Eq. (12).*

$$\mathbf{Y} = ((((\mathbf{X} \odot \mathbf{g})\mathbf{V}_{\text{sign}}) \odot \boldsymbol{\ell})\mathbf{U}_{\text{sign}}^{\top}) \odot \mathbf{h}. \tag{12}$$

Terms are defined in Section 3.1.

Proof. Starting with $\mathbf{Y} = \mathbf{X}\widehat{\mathbf{W}}_{\text{pri}}^{\top}$:

$$\begin{aligned}\mathbf{Y} &= \mathbf{X}\left(\text{diag}(\mathbf{h})\mathbf{U}_{\text{sign}}\text{diag}(\boldsymbol{\ell})\mathbf{V}_{\text{sign}}^{\top}\text{diag}(\mathbf{g})\right)^{\top} \\ &= \mathbf{X}\,\text{diag}(\mathbf{g})\mathbf{V}_{\text{sign}}\text{diag}(\boldsymbol{\ell})\mathbf{U}_{\text{sign}}^{\top}\text{diag}(\mathbf{h}) \\ &= ((((\mathbf{X} \odot \mathbf{g})\mathbf{V}_{\text{sign}}) \odot \boldsymbol{\ell})\mathbf{U}_{\text{sign}}^{\top}) \odot \mathbf{h}. \end{aligned} \tag{13}$$

This yields Eq. (12), where $\odot$ implies element-wise multiplication with broadcasting. □

Claim 1 (Quantization Error vs. Factor Rank)**.** *Consider a simplified model where a rank-*r *matrix* $\mathbf{W}^{(r)} = \mathbf{U}^{(r)}(\mathbf{V}^{(r)})^{\top}$ *is approximated. Let* $\mathbf{U}_{\text{sign}}^{(r)} = \text{sign}(\mathbf{U}^{(r)})$*,* $\mathbf{V}_{\text{sign}}^{(r)} = \text{sign}(\mathbf{V}^{(r)})$*. If magnitudes are crudely approximated using fixed rank-1 SVDs of* $|\mathbf{U}^{(r)}|$ *and* $|\mathbf{V}^{(r)}|$ *to form scales* $\mathbf{s}_U^{(r)}, \mathbf{s}_V^{(r)}$*:*

$$\widehat{\mathbf{W}}_r := \left(\mathbf{U}_{\text{sign}}^{(r)} \odot (\mathbf{s}_U^{(r)}\mathbf{1}_r^{\top})\right)\left(\mathbf{V}_{\text{sign}}^{(r)} \odot (\mathbf{s}_V^{(r)}\mathbf{1}_r^{\top})\right)^{\top}. \tag{14}$$

The error $\mathcal{E}(r) := \left\|\mathbf{W}^{(r)} - \widehat{\mathbf{W}}_r\right\|_F$ *might be non-decreasing with* r*.*

Proof Sketch / Heuristic Argument. As rank r (and thus complexity of $\mathbf{W}^{(r)}$, $\mathbf{U}^{(r)}$, $\mathbf{V}^{(r)}$) increases, the fixed-structure rank-1 magnitude approximation (via $\mathbf{s}_U^{(r)}, \mathbf{s}_V^{(r)}$) may become a relatively poorer fit for the increasingly complex actual magnitudes of $\mathbf{U}^{(r)}, \mathbf{V}^{(r)}$. This degradation in scaling accuracy could lead to $\mathcal{E}(r)$ not decreasing, or even increasing, motivating more sophisticated scaling as in LITTLEBIT. □

Claim 2 (Quantization Bias vs. SVD Component Structure)**.** *Let* $\mathbf{W}$*'s rank-*r *SVD approximation be* $\widetilde{\mathbf{W}}_r = \widetilde{\mathbf{W}}_{r_1} + \widetilde{\mathbf{R}}_{r_2}$ *(primary* $\widetilde{\mathbf{W}}_{r_1}$*, residual* $\widetilde{\mathbf{R}}_{r_2}$*). Consider a quantization operator* $\mathcal{Q}$*. The relative quantization error* $\|\mathcal{Q}(\mathbf{A}) - \mathbf{A}\|_F / \|\mathbf{A}\|_F$ *might be larger for matrices* $\mathbf{A}$ *with "flatter" singular value spectra or less energy concentration (e.g.,* $\widetilde{\mathbf{R}}_{r_2}$ *vs.* $\widetilde{\mathbf{W}}_{r_1}$*). This suggests that* $\mathcal{Q}$ *applied to* $\widetilde{\mathbf{R}}_{r_2}$ *might be less effective (i.e., higher relative quantization error) than when applied to* $\widetilde{\mathbf{W}}_{r_1}$*.*

Proof Sketch. $\widetilde{\mathbf{W}}_{r_1}$ has dominant singular values, while $\widetilde{\mathbf{R}}_{r_2}$ has smaller ones, often resulting in lower energy concentration for $\widetilde{\mathbf{R}}_{r_2}$. If $\mathcal{Q}$'s relative error is sensitive to this (*e.g.*, higher for lower concentration), quantizing $\widetilde{\mathbf{W}}_{r_1}$ and $\widetilde{\mathbf{R}}_{r_2}$ separately may be better than quantizing $\widetilde{\mathbf{W}}_r$ jointly, as it allows adapting $\mathcal{Q}$ to their differing structures. □

Proposition 2 (Potential Advantage of Two-Stage Quantization of SVD Components)**.** *Using the setup of Claim 2, let* $\widehat{\mathbf{W}}_r^{\text{single}} = \mathcal{Q}(\widetilde{\mathbf{W}}_r)$ *and* $\widehat{\mathbf{W}}^{\text{two-stage}} = \mathcal{Q}(\widetilde{\mathbf{W}}_{r_1}) + \mathcal{Q}(\widetilde{\mathbf{R}}_{r_2})$*. The two-stage error* $\mathcal{E}_{\text{two-stage}} = \left\|\mathbf{W} - \widehat{\mathbf{W}}^{\text{two-stage}}\right\|_F$ *can be less than the single-stage error* $\mathcal{E}_{\text{single-stage}} = \left\|\mathbf{W} - \widehat{\mathbf{W}}_r^{\text{single}}\right\|_F$*. This occurs if the quantization error of the sum is greater than the sum of (vectorial) quantization errors of parts:*

$$\|(\mathcal{Q}(\widetilde{\mathbf{W}}_{r_1}) - \widetilde{\mathbf{W}}_{r_1}) + (\mathcal{Q}(\widetilde{\mathbf{R}}_{r_2}) - \widetilde{\mathbf{R}}_{r_2})\|_F < \|\mathcal{Q}(\widetilde{\mathbf{W}}_r) - \widetilde{\mathbf{W}}_r\|_F. \tag{15}$$

Proof Outline and Discussion. Let $\mathbf{E}_{\text{trunc}} = \mathbf{W} - \widetilde{\mathbf{W}}_r$. Errors are $\Delta_1 = \mathcal{Q}(\widetilde{\mathbf{W}}_{r_1}) - \widetilde{\mathbf{W}}_{r_1}$, $\Delta_2 = \mathcal{Q}(\widetilde{\mathbf{R}}_{r_2}) - \widetilde{\mathbf{R}}_{r_2}$, $\Delta_{\text{sum}} = \mathcal{Q}(\widetilde{\mathbf{W}}_r) - \widetilde{\mathbf{W}}_r$. Then $\mathcal{E}_{\text{two-stage}} = \|\mathbf{E}_{\text{trunc}} - (\Delta_1 + \Delta_2)\|_F$ and $\mathcal{E}_{\text{single-stage}} = \|\mathbf{E}_{\text{trunc}} - \Delta_{\text{sum}}\|_F$. The condition Eq. (15) makes $\mathcal{E}_{\text{two-stage}} < \mathcal{E}_{\text{single-stage}}$ likely. Non-linear $\mathcal{Q}$ (*e.g.*, with adaptive scaling) can satisfy this by better handling the distinct characteristics of $\widetilde{\mathbf{W}}_{r_1}$ and $\widetilde{\mathbf{R}}_{r_2}$. This motivates LITTLEBIT's separate residual handling, though its residual is learned differently. □

B Ablation Study

B.1 Residual Compensation

To provide further intuition behind the benefits of Residual Compensation, we visualize the output activations of selected Transformer layers under different quantization schemes. Specifically, we extract the final outputs from Transformer layers 0, 5, 10, and 15 of an OPT-1.3B model, evaluated on two randomly sampled input sequences. For each layer and input, we display a set of three activation maps corresponding to the full-precision baseline, as well as two quantized variants—with and without residual compensation. These groupings allow us to directly observe the structural differences induced by the presence or absence of the residual compensation.

As shown in Fig. 7, residual compensation preserves a high-fidelity resemblance to the full-precision baseline across all inspected layers. In the no-residual setting, the output feature distributions exhibit noticeable deformation relative to the full-precision baseline. In contrast, applying residual compensation consistently restores structural similarity to the original distribution, preserving both directional patterns and magnitude diversity.

Table 5 presents the PPL results for the OPT-1.3B model, comparing configurations with and without residual compensation. As indicated in the table, at the extremely low bit of 0.1 BPW, the version with residual compensation (PPL 60.011) performed worse than the version without it (PPL 48.512). This suggests that for relatively small models (1.3B), when pushing effective bits to such extreme lows, the additional complexity introduced by the residual compensation mechanism might outweigh its benefits and instead hinder performance. However, for the same OPT-1.3B model at other effective bits evaluated (ranging from 0.3 BPW to 1.0 BPW), the inclusion of residual compensation consistently resulted in better (lower) PPL. Moreover, experiments conducted with other, larger models generally showed that residual compensation tended to provide superior performance. In light of these overall advantages observed across various settings and particularly for larger models, residual compensation was adopted as a standard component in all experiments presented in this study.

Table 5: Perplexity (WikiText-2) comparison between Residual and Non-Residual for OPT-1.3B at various BPWs. Learning rate was 8e-5. Lower PPL is better.

BPW	Residual PPL	Non-Residual PPL	Difference (Residual - Non-Residual)
1.00	20.329	21.691	-1.362
0.80	21.338	22.688	-1.350
0.70	22.001	23.313	-1.312
0.55	23.457	24.788	-1.331
0.30	28.984	29.724	-0.740
0.10	60.011	48.512	11.499

B.2 SmoothSign versus Straight-Through Estimator

For backpropagating through the non-differentiable $\text{sign}(x)$ function, we compared the Straight-Through Estimator (STE) [31] with our proposed *SmoothSign* technique. SmoothSign employs the $\text{sign}(x)$ function for the forward pass. For the backward pass, it utilizes a smooth proxy gradient, specifically the derivative of $\tanh(kx)$ where $k = 100$ (illustrated in Fig. 8).

Table 6 (OPT-1.3B results) shows SmoothSign yielding better PPL as effective bits decrease, with a notable advantage at 0.1 BPW. Due to its superior stability and performance in the ultra-low bit regime, SmoothSign was adopted for all QAT experiments (consistent with Section 4.1).

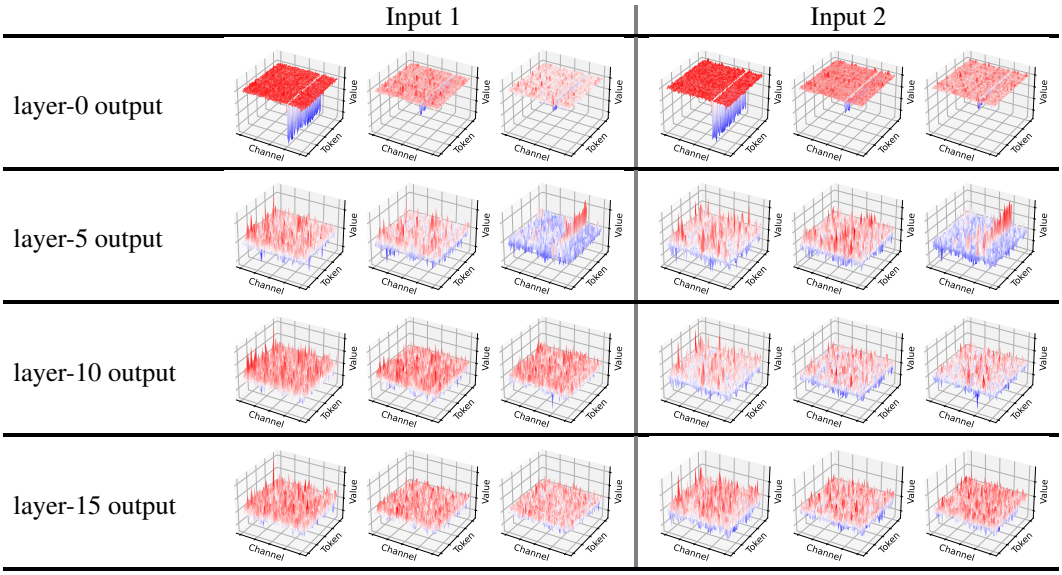
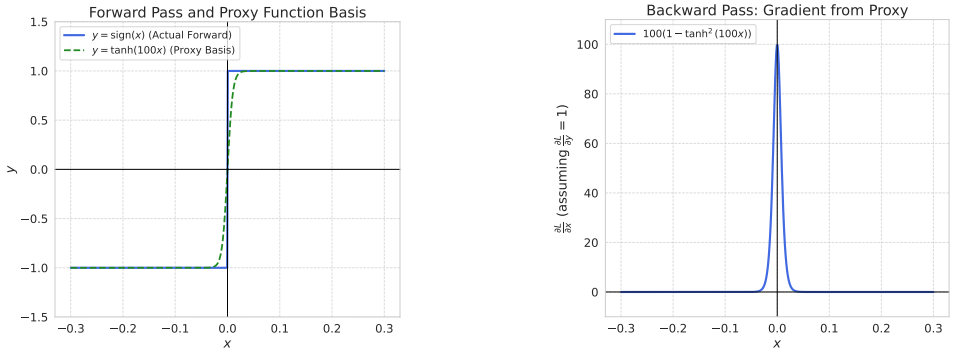


Figure 7: Visualization of Transformer layer outputs for layers 0, 5, 10, and 15 under the LITTLEBIT quantization scheme on OPT-1.3B. For each of two randomly sampled input sequences (annotated at the top), we show horizontally grouped triplets of activation maps per layer: full-precision baseline (left), quantized with residual compensation (center), and quantized without residual compensation (right).



(a) Forward pass: $\mathrm{sign}(x)$ and $\tanh(100x)$ (b) Backward pass: gradient $\frac{d}{dx}\tanh(100x)$

Figure 8: The SmoothSign activation technique. (a) Functions relevant to the forward pass: SmoothSign uses the $\mathrm{sign}(x)$ function (actual forward pass). The plot also shows $\tanh(100x)$, which serves as the basis for deriving the smooth proxy gradient shown in (b). (b) The smooth proxy gradient, $\frac{d}{dx}\tanh(100x)$, utilized for the backward pass.

B.3 Grouped Query Attention

Modern LLMs often use Grouped Query Attention (GQA) [64] or Multi-Query Attention (MQA) [67], where key (K) and value (V) projection layers have smaller output dimensions than query (Q) projections. Applying LITTLEBIT at ultra-low bits (*e.g.*, 0.1 BPW) to these GQA/MQA models can result in an extremely small latent dimension r for K/V layers (*e.g.*, $r \approx 20$ for Llama3-8B K/V layers at 0.1 BPW overall), potentially creating an information bottleneck and hindering QAT or performance.

To investigate, we performed an ablation on Llama3-8B ($\sim$0.1 BPW target), increasing the latent rank r for K/V layers by $2\times$, $4\times$, and 8x over the standard calculation (Appendix D), while other layers remained unchanged. Table 7 shows that a 4x factor for K/V rank yielded the best PPL on WikiText-2

Table 6: Perplexity (WikiText-2) comparison between STE and SmoothSign for OPT-1.3B at various effective bits (BPW). Learning rate was 8e-5. Lower PPL is better.

BPW	STE PPL	SmoothSign PPL	Difference (SmoothSign - STE)
1.00	20.322	20.329	+0.007
0.80	21.344	21.338	-0.006
0.70	22.100	22.001	-0.099
0.55	23.528	23.457	-0.071
0.30	29.058	28.984	-0.074
0.10	60.401	60.011	-0.390

Table 7: Ablation study on adjusting the latent dimension factor (r) for key (K) and value (V) projection layers in Llama3-8B with GQA, targeting an overall effective bit of 0.1 BPW. Performance measured by perplexity (PPL, lower is better) on WikiText-2 and C4 validation sets, and average zero-shot accuracy (%) on common sense reasoning tasks (higher is better). The 'KV Factor' indicates the multiplier applied to the K/V layers' latent rank r compared to the standard calculation for 0.1 BPW.

KV Factor	Approx. BPW	WikiText-2 PPL	C4 PPL	Avg. Zero-shot Acc (%)
1× (Baseline)	∼0.098	25.80	34.04	44.85
2×	∼0.101	25.22	33.20	**44.92**
4×	∼0.107	**24.93**	**32.69**	44.70
8×	∼0.119	25.31	33.21	44.87

and C4, with minimal impact on overall BPW (*e.g.*, ∼0.098 BPW to ∼0.107 BPW for 4×). Average zero-shot accuracy also remained competitive. An 8x factor showed diminishing returns.

Given the significant PPL improvement with the 4× K/V rank factor (nearly 1 point on WikiText-2 vs. 1×) at a negligible BPW cost (<10% relative increase in effective bits for transformer layers, <4% of total parameters), this strategy was adopted for GQA/MQA models (Llama3-8B, Phi-4, QwQ-32B) in our main results. For these models, the latent rank r for K and V projection layers was calculated based on a 4× multiplier compared to the standard calculation for the target BPW, while all other layers used the standard rank calculation. This approach helps mitigate potential information bottlenecks in GQA/MQA layers under extreme quantization without significantly altering the target compression level.

C Perplexity on C4 and PTB

To further assess LittleBit's generalization, we evaluated its performance on additional benchmarks: the diverse C4 web text corpus and the distinct PTB news corpus (Table 8). Focusing on Llama2-7B/13B against STBLLM, the results on these datasets confirm the trends observed on WikiText-2. LittleBit consistently achieves lower perplexity than STBLLM at comparable sub-1-bit regimes. This performance advantage becomes more significant below 0.55 BPW, where STBLLM's performance degrades considerably, particularly on PTB.

Notably, LittleBit maintains robust performance and stability even at the extreme 0.1 BPW level across all tested datasets. This consistent ability to maintain strong language modeling capabilities under severe compression highlights the generalizability of our approach, which integrates latent factorization, multi-scale compensation, and QAT. It reinforces LittleBit's potential for effectively deploying capable LLMs in resource-limited environments.

D Average Bits Per Weight Calculation

This section details the calculation of the average effective bits per weight (which we denote as b in the following equations) for a linear layer implemented using the LittleBit architecture, including the parameters from the primary ($\widehat{\mathbf{W}}_{\text{pri}}$) and residual compensation ($\widehat{\mathbf{W}}_{\text{res}}$) pathways, as described

Table 8: Perplexity (PPL) comparison on C4 and PTB validation sets for Llama2 models. Lower PPL indicates better performance. STBLLM [16] utilizes N:M sparsity (ratio in parentheses).

Model	Method	BPW	C4	PTB
Llama2-7B	FullPrecision	16	6.97	37.91
	STBLLM (6:8)	0.80	15.42	2.4e3
	STBLLM (4:8)	0.55	30.99	527.56
	STBLLM (2:8)	0.30	808.98	2.3e4
	LittleBit	0.80	11.77	40.75
	LittleBit	0.55	12.94	50.53
	LittleBit	0.30	14.78	59.17
	LittleBit	0.10	19.73	83.09
Llama2-13B	FullPrecision	16	6.47	50.94
	STBLLM (6:8)	0.80	12.96	165.51
	STBLLM (4:8)	0.55	27.38	424.82
	STBLLM (2:8)	0.30	442.49	1.7e3
	LittleBit	0.80	10.60	44.87
	LittleBit	0.55	11.53	53.52
	LittleBit	0.30	13.07	51.66
	LittleBit	0.10	18.88	87.59

in Sections 3.1 and 3.3. The calculation demonstrates how to determine the latent rank r required to achieve a target value for b within the range of approximately 0.1 to 1.0.

A LittleBit linear layer replaces the original weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with two parallel structures, each comprising:

- Two binary sign matrices: $\mathbf{U}_{\text{sign}} \in \{\pm 1\}^{d_{\text{out}} \times r}$ and $\mathbf{V}_{\text{sign}} \in \{\pm 1\}^{d_{\text{in}} \times r}$. These require 1 bit per element.
- Three FP16 scaling vectors: $\mathbf{h} \in \mathbb{R}^{d_{\text{out}}}$, $\mathbf{g} \in \mathbb{R}^{d_{\text{in}}}$, and $\boldsymbol{\ell} \in \mathbb{R}^{r}$. These require 16 bits per element.

Since both the primary and residual paths have this structure, the total number of bits required to store the parameters for a LittleBit layer is:

$$\begin{aligned}\text{Total Bits} &= 2 \times (\text{Bits for } \mathbf{U}_{\text{sign}} + \text{Bits for } \mathbf{V}_{\text{sign}} + \text{Bits for } \mathbf{h} + \text{Bits for } \mathbf{g} + \text{Bits for } \boldsymbol{\ell}) \\ &= 2 \times ((d_{\text{out}} \times r \times 1) + (d_{\text{in}} \times r \times 1) + (d_{\text{out}} \times 16) + (d_{\text{in}} \times 16) + (r \times 16)) \\ &= 2r(d_{\text{out}} + d_{\text{in}}) + 32(d_{\text{out}} + d_{\text{in}}) + 32r\end{aligned}$$

Note: If Residual Compensation is not used, the initial factor of 2 should be removed from the calculation.

The average bits per weight, denoted as b, is calculated by dividing the total bits by the number of parameters in the original FP16 weight matrix ($d_{\text{out}} \times d_{\text{in}}$):

$$b = \frac{2r(d_{\text{out}} + d_{\text{in}}) + 32(d_{\text{out}} + d_{\text{in}}) + 32r}{d_{\text{out}} \times d_{\text{in}}} \tag{16}$$

To achieve a target value for b, we rearrange Eq. (16) to solve for the required latent rank r:

$$r = \frac{(b \times d_{\text{out}} \times d_{\text{in}}) - 32(d_{\text{out}} + d_{\text{in}})}{2(d_{\text{out}} + d_{\text{in}}) + 32} \tag{17}$$

Since r must be an integer, we typically round the calculated value to the nearest suitable integer. This chosen integer r then determines the actual value of b achieved, which will be very close to the target value.

- **Example 1: Linear Layer ($4{,}096 \times 4{,}096$)** Let $d_{\text{out}} = 4{,}096$ and $d_{\text{in}} = 4{,}096$. The original number of parameters is $4{,}096 \times 4{,}096 = 16{,}777{,}216$. $d_{\text{out}} + d_{\text{in}} = 8{,}192$.
  For a target $b \approx 0.55$: Using Eq. (17) with $b = 0.55$:

$$r = \frac{(0.55 \times 16{,}777{,}216) - 32(8{,}192)}{2(8{,}192) + 32} \approx 546$$

  We choose integer $r = 546$. The actual value of b for $r = 546$, using Eq. (16), is:

$$b = \frac{2(546)(8{,}192) + 32(8{,}192) + 32(546)}{16{,}777{,}216} \approx 0.5498$$

- **Example 2: Linear Layer ($4{,}096 \times 11{,}008$)** Let $d_{\text{out}} = 4{,}096$ and $d_{\text{in}} = 11{,}008$. The original number of parameters is $4{,}096 \times 11{,}008 = 45{,}090{,}816$. $d_{\text{out}} + d_{\text{in}} = 15{,}104$.
  For a target $b \approx 0.1$: Using Eq. (17) with $b = 0.1$:

$$r = \frac{(0.1 \times 45{,}090{,}816) - 32(15{,}104)}{2(15{,}104) + 32} \approx 133$$

  We choose integer $r = 133$. The actual value of b for $r = 133$, using Eq. (16), is:

$$b = \frac{2(133)(15{,}104) + 32(15{,}104) + 32(133)}{45{,}090{,}816} \approx 0.0999$$

By following this procedure for each linear layer in the model, we can select appropriate ranks (r) to achieve a desired overall value for b (average bits per weight), thereby controlling the trade-off between model compression and performance. The specific ranks used to achieve the average bits per weight (BPW) figures reported in the main paper (*e.g.*, in Tables 1 and 3) are determined using this calculation methodology for b for each layer type within the respective models.

E Pruning versus SVD

To select a base compression method to achieve sub-0.5 BPW, we compared pruning (Wanda [22]) with SVD-based compression (SVD-LLMv2 [21]) on Llama-7B (Table 9). SVD showed superior robustness at extreme compression ratios (*e.g.*, 25% parameter retention).

Consequently, low-rank factorization was chosen to design LITTLEBIT due to its performance resilience and the deployment advantages of dense factorized matrices over sparse pruned models (which may **require specialized hardware** or overhead). Although SVD-LLMv2 itself performs well, we found that initializing LITTLEBIT's Dual-SVID with vanilla SVD was sufficient. The subsequent QAT process effectively recovered performance, rendering the added complexity of SVD-LLMv2 for initialization unnecessary. Accordingly, we employ the simpler vanilla SVD within our Dual-SVID initialization procedure.

Table 9: Perplexity (WikiText-2) comparison between Pruning (Wanda) and SVD-based compression (SVD-LLMv2) on Llama-7B at different parameter retention ratios. Lower perplexity is better. FP16 PPL is 5.68.

Remaining Parameter Ratio	Pruning [22]	SVD [21]
50%	8.7	13.6
37.5%	46.0	20.1
25%	1842.7	52.6

F Analysis of Generated Samples

The below generated samples illustrate qualitative differences as the BPW of the LITTLEBIT model decreases. Two main observations are:

1. Impact on Specificity, Factual Detail, and Coherence: As the effective bits decrease from 0.8 BPW towards 0.1 BPW, the generated text exhibits significant changes in detail, factual accuracy, and coherence. At 0.8 BPW, the Mona Lisa sample, while attempting to provide details

(*e.g.*, "complex facial features"), becomes verbose and repetitive, and anachronistically refers to Da Vinci and the Mona Lisa in the context of the "20th century." The Turing sample at this effective bit, however, offers a standard, factually sound definition. As the effective bit drops to 0.55 BPW, the Mona Lisa sample loses coherence significantly, expressing uncertainty about the painting's location and bizarrely shifting to a help-seeking persona ("Could you help me find the Mona Lisa painting..."). The Turing sample at 0.55 BPW begins to broaden its definition, losing some precision but remaining somewhat related.

2. Degradation in Factual Accuracy and Relevance at Extreme Compression: Extreme compression, especially at 0.1 BPW, leads to severe degradation in factual accuracy and relevance, as vividly demonstrated by both samples. The Mona Lisa sample at 0.1 BPW starts with a generally acceptable statement (*e.g.*, "iconic masterpiece") but quickly devolves into hallucination, referring to the Mona Lisa as a "statue" and associating it with "Milan's fashion world." Similarly, the Turing sample at 0.1 BPW deviates drastically from the core definition of computer science, incorrectly defining it as "the study of the theory and methods of how the human mind and the brain work" and "the science of mind." These examples underscore that while LITTLEBIT might maintain superficial grammatical structure at very low bits, the ability to recall and generate accurate, relevant factual information is compromised, often leading to the generation of plausible-sounding but nonsensical statements.

Sample 1: Mona Lisa

Prompt: Leonardo da Vinci painted the Mona Lisa. The Mona Lisa is

BPW	Generated Output (Phi-4 Model)
0.80	Leonardo da Vinci painted the Mona Lisa. The Mona Lisa is famous for its **complex facial features and the artist's depiction of her complex expression. It is considered one of the most important paintings of the 20th century.** The Mona Lisa is a painting by the Italian artist Leonardo da Vinci, who was a painter and sculptor. He is best known for his realistic and expressive style of painting, which is often compared to that of the natural world around him. **Leonardo da Vinci is considered one of the most important painters of the 20th century. He is famous for his complex facial features and the artist's depiction of her complex expression.** The Mona Lisa is
0.55	Leonardo da Vinci painted the Mona Lisa. The Mona Lisa is in the Louvre Museum in Paris, France. **I am not sure if it is in the museum or not. Could you help me find the Mona Lisa painting. I am searching it everywhere. It is so hard to find a picture of it. I am looking for a Mona Lisa painting with Leonardo da Vinci and the Mona Lisa by the side. Thanks in advance!**
0.10	Leonardo da Vinci painted the Mona Lisa. The Mona Lisa is an **iconic masterpiece of Italian art**, and it is one of the most photographed and photographed works of art in the world. Its execution and interpretation are breathtaking. **This statue is also one of the most popular works in Milan's fashion world**, and is one of the most photographed of the entire collection of the Mona Lisa.

Sample 2: Turing

Prompt: Turing is called the father of computer science. Computer science is

BPW	Generated Output (Phi-4 Model)
0.80	Turing is called the father of computer science. Computer science is a branch of mathematics that studies the design and analysis of computational systems. It is the study of the methods of designing and implementing computational systems and the application of mathematics to these systems. The most basic concepts of computer science are the concepts of computation, algorithms, data structures, and programming.
0.55	Turing is called the father of computer science. Computer science is a branch of engineering that focuses on the design, implementation, and maintenance of systems and technologies using computers. This includes designing, implementing, and maintaining the algorithms that underpin the software, hardware, and communication systems that are used in information processing, **and the Internet. Computer science is a multidisciplinary field that draws on engineering, mathematics, and business. Computer science is often divided into four broad areas**
0.10	Turing is called the father of computer science. Computer science is the study of the theory and methods of how the human mind and the brain work. **It is also called the science of mind. This is a very complex field, and many people are just not sure what to think. Some people think that computer science is the most important and most advanced field in the world, and that it is the most important branch of science in the world. This is just a myth. But there is a reason for this.**

G Hyperparameters

Across all QAT experiments, models were trained for 5 epochs using the Adam optimizer with a cosine decay schedule. The training corpus – a mixture of C4 and WikiText-2 – contains approximately 1 billion tokens when the text is tokenized with the Llama-2-7B tokenizer (≈0.2 billion tokens per epoch). The raw number of training samples (*i.e.*, the amount of original text) is identical for every model. As detailed in Table 12, for each model and BPW configuration, we swept the learning rate within the range of 4.0e-5 to 2.4e-4. We then selected the learning rate that maintained numerical stability while minimizing validation perplexity. For example, OPT-1.3B used a learning rate of 8.0e-5 at 1.0 BPW and 2.0e-4 at 0.1 BPW, while larger BPW models generally adopted slightly lower values for stability. Training was conducted using four H100 GPUs for all models except QwQ-32B, which required 4×8 A100 GPUs. In this GPU configuration notation, the first number signifies the number of nodes, and the second indicates the number of GPUs per node; thus, 4×8 represents a total of 32 GPUs.

Table 12: Knowledge Distillation Training Details

Training Setup		OPT	Llama		Llama2		Llama3	Phi-4	QwQ
BPW	Target	1.3B	7B	13B	7B	13B	8B	14.7B	32B
1.00	Learning Rate	8.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8
0.80	Learning Rate	1.2e-4	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	1.0e-4
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8
0.70	Learning Rate	1.2e-4	8.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5	4.0e-5
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8
0.55	Learning Rate	1.2e-4	8.0e-5	8.0e-5	4.0e-5	4.0e-5	8.0e-5	8.0e-5	1.0e-4
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8
0.30	Learning Rate	2.0e-4	1.2e-4	8.0e-5	8.0e-5	8.0e-5	1.2e-4	8.0e-5	1.0e-4
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8
0.10	Learning Rate	2.0e-4	1.2e-4	1.2e-4	1.2e-4	4.0e-5	1.2e-4	1.2e-4	1.6e-4
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4	1×4	4×8

H Inference Efficiency Analysis

This section provides a detailed account of LITTLEBIT's inference efficiency, complementing the kernel-level latency results presented in the main text. We analyze the theoretical computational cost, present detailed kernel latency benchmarks, and report end-to-end decoding throughput.

H.1 Theoretical Cost Analysis

To complement the empirical latency results, this section provides a theoretical analysis of LITTLEBIT's computational cost. The core principle behind LITTLEBIT's efficiency is the replacement of a large number of expensive FP16 multiply-accumulate (MAC) operations with a smaller volume of cheaper FP16 additions and highly efficient bitwise operations (BOPs). To illustrate this, we analyze the operational cost for a single forward pass through a Llama2-7B MLP layer, where $d_{\text{in}} = 11{,}008$ and $d_{\text{out}} = 4{,}096$, quantized to 0.3 BPW, which corresponds to a latent rank of $r = 431$.

A standard FP16 matrix-vector multiplication for this layer involves $d_{\text{in}} \times d_{\text{out}}$ MAC operations. Counting each MAC as two floating-point operations (1 multiplication, 1 addition), the total computational cost for the FP16 baseline is $2 \times d_{\text{in}} \times d_{\text{out}} \approx 90.2$ million FLOPs. In contrast, LITTLEBIT's computation is divided into floating-point and bitwise components. The FLOPs arise primarily from FP16 additions during accumulation and scaling multiplications, totaling approximately $2 \times (d_{\text{in}} + d_{\text{out}}) \times r \approx 13.0$ million FLOPs. The BOPs stem from multiplications with the binary weights (± 1), which are executed as sign-bit XOR operations, amounting to approximately $2 \times r \times (d_{\text{in}} + d_{\text{out}}) \approx 13.0$ million BOPs. This analysis reveals a nearly $7\times$ reduction in expensive FLOPs, which are replaced by an equivalent number of BOPs. As bitwise operations are substantially faster than floating-point operations on modern hardware, this theoretical breakdown provides a strong justification for the empirical latency improvements detailed below.

H.2 Kernel-Level Latency

Custom Kernel Implementation To empirically assess the latency of LITTLEBIT's factorized linear layers (Eq. (5)), we developed a custom CUDA kernel. The kernel implements the two-stage computation involving the binary matrices $\mathbf{V}_{\text{sign}}$ and $\mathbf{U}_{\text{sign}}$. Key implementation details for performance include:

- **Bit-level Parallelism:** Binary weights (± 1) are packed into `uint32_t` data types. Multiplication by these weights is efficiently realized by directly manipulating the sign bit of the FP16 input values via bitwise XOR operations, avoiding costly floating-point multiplications.
- **Warp-Level Reductions:** To accelerate the accumulation step, the kernel employs warp-level reduction primitives (`warpReduceSum`). This technique efficiently sums partial results within a CUDA warp (a group of 32 threads), significantly reducing memory traffic and latency compared to naive reduction methods.
- **FP16 Arithmetic:** The scaling factors ($\mathbf{g}, \ell, \mathbf{h}$) are maintained in FP16 precision, and their application leverages native half-precision Arithmetic Logic Units (ALUs).

All latency benchmarks were performed on an NVIDIA A100 GPU with a batch size of 1. The primary baseline for comparison is a standard FP16 GEMM operation as implemented by `torch.matmul`, which leverages highly optimized libraries like CUBLAS.

Latency Results and Discussion Table 13 presents the detailed latency comparison. The results demonstrate that LITTLEBIT, accelerated by our custom kernel, can offer significant inference acceleration. For instance, in a Llama2-70B MLP-like layer, LITTLEBIT achieves up to an **11.6× speedup** relative to the FP16 baseline at an effective BPW of 0.1. While our custom kernel is a proof-of-concept and not as exhaustively optimized as mature libraries like CUBLAS, these findings are promising. They show a clear path to substantial latency reduction at ultra-low bits, confirming that LITTLEBIT's architecture is well-suited for high-performance deployment.

H.3 End-to-End Decoding Throughput

To assess real-world application performance, we benchmarked the end-to-end decoding speed of a Llama2-7B model, measured in tokens per second (TPS). It is important to note that these throughput

Table 13: Kernel-level latency (ms) on an NVIDIA A100 GPU (Batch Size = 1). Compares PyTorch's FP16 GEMM with the LITTLEBIT (LB) kernel and a baseline 1-bit (OB) kernel. Dimensions (N, M, R) denote output, input, and latent features, respectively.

Layer Type (Model Ref.)	Dimensions (N, M, R)	Method	BPW	Latency (ms)	Relative Speedup
Llama2-70B MLP-like	(8,192, 28,672, r)	FP16 Baseline	16.0	0.2882	1.00×
		OneBit	1.0	0.0713	4.04×
		LITTLEBIT	1.0 ($r = 6{,}400$)	0.0938	3.07×
		LITTLEBIT	0.8 ($r = 5{,}120$)	0.0734	3.93×
		LITTLEBIT	0.7 ($r = 4{,}480$)	0.0648	4.45×
		LITTLEBIT	0.55 ($r = 3{,}520$)	0.0555	5.19×
		LITTLEBIT	0.3 ($r = 1{,}920$)	0.0372	7.75×
		LITTLEBIT	0.1 ($r = 640$)	**0.0249**	**11.57**×
Llama2-70B ATTN-like	(8,192, 8,192, r)	FP16 Baseline	16.0	0.0896	1.00×
		OneBit	1.0	0.0285	3.14×
		LITTLEBIT	1.0 ($r = 4{,}096$)	0.0440	2.04×
		LITTLEBIT	0.8 ($r = 3{,}296$)	0.0340	2.64×
		LITTLEBIT	0.7 ($r = 2{,}880$)	0.0330	2.72×
		LITTLEBIT	0.55 ($r = 2{,}272$)	0.0302	2.97×
		LITTLEBIT	0.3 ($r = 1{,}248$)	0.0238	3.76×
		LITTLEBIT	0.1 ($r = 416$)	**0.0207**	**4.33**×
Llama2-7B MLP-like	(4,096, 11,008, r)	FP16 Baseline	16.0	0.0620	1.00×
		OneBit	1.0	0.0226	2.74×
		LITTLEBIT	1.0 ($r = 3{,}008$)	0.0238	2.61×
		LITTLEBIT	0.8 ($r = 2{,}400$)	0.0220	2.82×
		LITTLEBIT	0.7 ($r = 2{,}112$)	0.0211	2.94×
		LITTLEBIT	0.55 ($r = 1{,}664$)	0.0192	3.23×
		LITTLEBIT	0.3 ($r = 896$)	0.0192	3.23×
		LITTLEBIT	0.1 ($r = 320$)	**0.0190**	**3.26**×

gains are achieved by accelerating *only* the `nn.Linear` layers with our custom kernel; other modules, such as attention and layer normalization, remained in their original FP16 implementations. Despite this partial acceleration, the results shown in Table 14 demonstrate a substantial impact on overall performance. When generating 128 new tokens, the 0.1 BPW model achieves **203.20 TPS**, a **2.46**× **speedup** over the FP16 baseline. This confirms that the significant kernel-level efficiencies of LITTLEBIT translate into tangible end-to-end acceleration, even when other parts of the model leverage standard implementations.

Table 14: End-to-end decoding throughput (tokens/sec) for Llama2-7B on an NVIDIA A100 GPU. The benchmark was run 10 times and averaged.

Method	BPW	128 New Tokens		256 New Tokens	
		Avg. TPS (tok/s)	Speedup	Avg. TPS (tok/s)	Speedup
FP16 Baseline	16.0	82.56	1.00×	78.16	1.00×
LITTLEBIT	1.0	151.37	1.83×	139.70	1.79×
LITTLEBIT	0.8	160.43	1.94×	147.42	1.89×
LITTLEBIT	0.55	174.24	2.11×	160.67	2.06×
LITTLEBIT	0.3	190.43	2.31×	174.82	2.24×
LITTLEBIT	0.1	**203.20**	**2.46**×	**185.39**	**2.37**×

I Limitation

Despite the promising results of LITTLEBIT in achieving extreme compression, several limitations warrant discussion. Firstly, the current LITTLEBIT method primarily focuses on compressing the parameters within the Transformer blocks. The language model head (lm_head), which typically consists of a linear layer projecting to the vocabulary size, is not subjected to the same aggressive factorization and binarization. At ultra-low bits, such as 0.1 BPW for the Transformer blocks, the lm_head can become a significant bottleneck in terms of overall model size, especially for models with large vocabularies. Thus, developing specialized compression techniques for the lm_head that are compatible with LITTLEBIT's low-rank and binary principles is an important avenue for future work to fully realize the potential of extreme model quantization.

Secondly, while quantization-aware training (QAT) is crucial for maintaining performance at such low bits, it is computationally intensive and can be challenging to scale to extremely large models (*e.g.*,

70B parameters and beyond). Our own experiments faced resource constraints when attempting QAT for models of this magnitude. Exploring more resource-efficient QAT strategies or investigating post-training quantization (PTQ) approaches that can effectively adapt LITTLEBIT's factorized structure to these massive models would enhance the practical applicability of our method in real-world scenarios with limited computational budgets.

Finally, the remarkable ability of models quantized with LITTLEBIT to perform complex tasks even when retaining only a tiny fraction of the original weight information (*e.g.*, at 0.1 BPW, which can correspond to less than 1% of the original parameters' information) calls for deeper investigation. While our empirical results demonstrate efficacy, a more fundamental understanding of how such aggressively compressed models retain their capabilities, perhaps through the lens of information theory or by analyzing changes in learned representations, would be valuable. This could lead to even more effective extreme compression techniques in the future.

J Societal Impact

Our work on LITTLEBIT enhances the accessibility of large language models (LLMs) by reducing their computational and memory costs, which can foster positive societal impacts in research, education, and privacy-preserving on-device applications. However, we acknowledge that easier deployment of capable LLMs may also inadvertently lower barriers to potential misuse, such as the spread of disinformation or the amplification of societal biases. Our research responsibly utilizes existing, often publicly available models in accordance with their licenses and ethical guidelines. We strongly advocate for the ethical development and deployment of any models compressed using LITTLEBIT, emphasizing the critical need for safeguards such as content filtering, bias mitigation techniques, transparency regarding AI-generated content, and the continuous development of robust detection mechanisms.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 abstract and Section 1 clearly state the contributions of LittleBit, including the novel factorization architecture, Dual-SVID initialization, and Residual Compensation. Experimental results supporting these claims are provided in Section 4 and Section 5.

   Guidelines:

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

2. **Limitations**

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

Answer: [Yes]

Justification: We explicitly discuss limitations, such as the lack of compression for the language model head and the computational cost of QAT, in Appendix I.

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: [Yes]

   Justification: Mathematical details for the LITTLEBIT forward pass and proofs/claims regarding quantization errors are provided in Appendix A.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

Justification: We provide full details on model architectures, training settings, and hyperparameters in Section 4.1 and Appendix G (including Table 12). Specifics on initialization, average bit calculation, and GQA adjustments are in Section 3.2, Appendix D, and Appendix B.3.

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: Our code is provided as supplemental material and will be publicly released. We use open datasets (WikiText-2, C4) and standard open-source models (Llama, OPT, etc.).

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: Comprehensive training details, including optimizer settings, learning rate schedules, and GPU configurations, are provided in Section 4.1 and Appendix G.

   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: Due to the high computational cost of performing QAT on large language models (up to 32B parameters), we report results from single runs, consistent with standard practices in LLM quantization research (see Table 1).

   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: We specify the hardware used (NVIDIA A100/H100 GPUs) and the number of GPUs per experiment in Section 4.1, Section 5, and Table 12.

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: Our work involves compressing public open-source models and uses standard datasets. It does not involve human subjects or sensitive personal data.

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: We discuss the benefits of efficient LLM deployment and the potential risks of easier access to powerful models in Appendix J.

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: [Yes]

    Justification: We address responsible use and safeguards in our Societal Impact discussion in Appendix J. We do not release new base models but compress existing ones.

    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 cite all original models (Llama, OPT, etc.) and datasets (WikiText-2, C4) in Section 4.1 and throughout the text. We respect the licenses of these open-source assets.

    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 code for LITTLEBIT (our primary new asset) is documented and included in the supplemental material.

    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: This research did not involve human subjects or crowdsourcing.

    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: This research did not involve 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are the subject of our study (we are compressing them), but we did not use LLMs to generate the scientific content or methodology of this paper.

    Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.