

BUTTERFLYQUANT: ULTRA-LOW-BIT LLM QUANTIZATION THROUGH LEARNABLE ORTHOGONAL BUTTERFLY TRANSFORMS

Anonymous authors

Paper under double-blind review

ABSTRACT

Large language models require massive memory footprints, severely limiting deployment on consumer hardware. Quantization reduces memory through lower numerical precision, but extreme 2-bit quantization suffers from catastrophic performance loss due to outliers in activations. Rotation-based methods such as QuIP and QuaRot apply orthogonal transforms to eliminate outliers before quantization, using computational invariance: $\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{Q}^T)(\mathbf{Q}\mathbf{x})$ for orthogonal \mathbf{Q} . However, these methods use fixed transforms—Hadamard matrices achieving optimal worst-case coherence $\mu = 1/\sqrt{n}$ —that cannot adapt to specific weight distributions. We identify that different transformer layers exhibit distinct outlier patterns, motivating layer-adaptive rotations rather than one-size-fits-all approaches. In this work, we propose ButterflyQuant, which replaces Hadamard rotations with learnable butterfly transforms parameterized by continuous Givens rotation angles. Unlike Hadamard’s discrete $\{+1, -1\}$ entries that are non-differentiable and thus prohibit gradient-based learning, butterfly transforms’ continuous parameterization enables smooth optimization while guaranteeing orthogonality by construction. This orthogonal constraint ensures theoretical guarantees in outlier suppression while achieving $O(n \log n)$ computational complexity with only $\frac{n \log n}{2}$ learnable parameters. We further introduce a uniformity regularization on post-transformation activations to promote smoother distributions amenable to quantization. Learning requires only 128 calibration samples and converges in minutes on a single GPU—a negligible one-time cost. For LLaMA-2-7B with 2-bit quantization, ButterflyQuant achieves 15.4 perplexity versus 37.3 for QuIP. The codes are in the Supplementary Materials.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities, but their deployment remains severely constrained by memory requirements. LLaMA-70B requires 140GB in FP16 precision, exceeding the capacity of most GPUs and making consumer deployment infeasible [Touvron et al., 2023, Zhao et al., 2025]. Recent research highlights deployment challenges including memory bandwidth bottlenecks, with inference serving as the dominant cost in production systems [Chen et al., 2024, Yuan et al., 2024]. Quantization—reducing numerical precision to 2-4 bits—offers a direct solution by compressing LLMs 4-8 \times . However, extreme quantization suffers from catastrophic performance degradation due to outliers in activations that dominate the dynamic range [Dettmers et al., 2022, Wei et al., 2022], a primary obstacle to low-bit compression [Sun et al., 2024].

To mitigate the outlier problem, rotation-based quantization methods have emerged as a robust solution [Ashkboos et al., 2024, Chee et al., 2023]. These methods apply an orthogonal transformation \mathbf{Q} before quantization, leveraging computational invariance: $\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{Q}^T)(\mathbf{Q}\mathbf{x})$. The rotation redistributes activations across channels, effectively smoothing out outlier features without altering the layer’s output. Prominent methods like QuaRot [Ashkboos et al., 2024] use fixed Hadamard transforms, which achieve optimal worst-case coherence, while QuIP [Chee et al., 2023] employs random orthogonal matrices. Despite their success, these approaches share a critical limitation: they apply a *fixed*, data-agnostic transform with discrete $\{+1, -1\}$ entries that cannot be optimized via gradients to all layers.

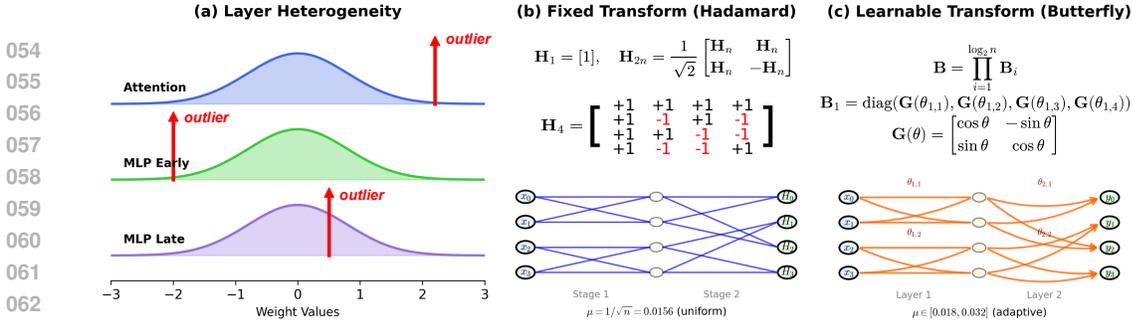


Figure 1: **Layer heterogeneity motivates learnable transforms for LLM quantization.** (a) Different transformer layers exhibit distinct outlier distributions: attention (positive tails), early MLP (negative regions), late MLP (boundaries). (b) Hadamard transforms with discrete $\{+1, -1\}$ entries apply fixed rotations through recursive decomposition $\mathbf{H}_{2n} = \frac{1}{\sqrt{2}}[\mathbf{H}_n, \mathbf{H}_n; \mathbf{H}_n, -\mathbf{H}_n]$, achieving uniform coherence $\mu = 1/\sqrt{n} = 0.0156$ across all layers. (c) Butterfly transforms use continuous rotation angles $\theta_{i,j}$ in Givens rotations $\mathbf{G}(\theta)$, enabling gradient-based optimization to learn layer-specific patterns. This yields adaptive coherence that matches each layer’s outlier distribution.

This one-size-fits-all strategy is fundamentally misaligned with the nature of LLMs, which exhibit significant *heterogeneity* across layers [Sun et al., 2024]. As illustrated in Figure 1(a), different transformer layers present unique quantization challenges: attention layers develop outliers in positive tails [Bondarenko et al., 2023], early MLP layers show them in negative regions [Wei et al., 2022], and late MLP layers have them near distribution boundaries [Sun et al., 2024]. These distinct patterns arise from varied architectural roles. Attention’s softmax operation naturally produces positive-skewed distributions [Bondarenko et al., 2023], early MLP gating functions (e.g., SwiGLU [Shazeer, 2020]) create asymmetric negative activations, while deeper layers accumulate numerical artifacts at distribution extremes [Sun et al., 2024, Dettmers et al., 2022]. This heterogeneity means that a single fixed rotation cannot be optimal for all layers. This layer-specific structure reveals a missed opportunity for optimization, motivating a shift from fixed to adaptive rotations.

To address this need for an adaptive yet efficient orthogonal transform, we propose **ButterflyQuant**. Our method replaces fixed Hadamard rotations with *learnable* butterfly transforms—structured orthogonal matrices factorized into $O(n \log n)$ Givens rotations [Dao et al., 2019]. As shown in Figure 1(c), butterfly transforms maintain the efficient computational structure of Hadamard matrices but learn layer-specific rotation angles via gradient descent. Crucially, butterfly transforms parameterize rotations through continuous angles $\theta \in \mathbb{R}$, enabling smooth gradient flow and stable optimization—in contrast to Hadamard’s discrete $\{+1, -1\}$ entries that prohibit gradient-based learning. This allows them to develop adaptive coherence patterns that are tailored to each layer’s unique outlier distribution. Unlike other learnable methods (e.g., SpinQuant [Liu et al., 2024a]) that optimize over the full Stiefel manifold with high computational cost, our sparse parameterization guarantees orthogonality by construction, enabling stable and efficient optimization. For non-power-of-2 dimensions common in LLMs (e.g., 5120), we develop composite transforms using Kronecker products. The learning process is remarkably lightweight, converging in minutes on a single GPU with a small calibration set.

2 RELATED WORK

2.1 POST-TRAINING QUANTIZATION FOR LLMs

Recent surveys [Zhao et al., 2025] categorize LLM PTQ methods into four main approaches: compensation, rotation, salience, and optimization-based techniques.

Compensation and Salience Methods. GPTQ [Frantar et al., 2023] uses second-order Hessian information for reconstruction error minimization. AWQ [Lin et al., 2024a] preserves salient weights while quantizing others based on activation magnitudes. SmoothQuant [Xiao et al., 2023] migrates difficulty from activations to weights via channel-wise scaling. OmniQuant [Shao et al., 2023] and AffineQuant [Guan et al., 2024a] apply learnable transformations for outlier suppression. Mixed-precision methods like LLM.int8() [Dettmers et al., 2022] and SpQR [Dettmers et al., 2023] maintain outlier channels in higher precision but increase complexity.

Optimization Methods. Recent approaches include VPTQ [Liu et al., 2024b] using vector quantization with optimized codebooks, APTQ [Guan et al., 2024b] with adaptive precision allocation, and AQLM [Egiazarian et al., 2024] employing additive quantization for 2-bit compression. While

108 achieving strong results, these methods require extensive optimization and lack theoretical guaran-
109 tees of rotation-based approaches.

111 2.2 ROTATION-BASED QUANTIZATION

112
113 Rotation-based methods eliminate outliers through orthogonal transformations leveraging computa-
114 tional invariance: for orthogonal \mathbf{Q} , $\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{Q}^T)(\mathbf{Q}\mathbf{x})$.

115 **Fixed Methods.** QuaRot [Ashkboos et al., 2024] applies Hadamard transforms to redistribute out-
116 liers. QuIP [Chee et al., 2023] and QuIP# [Tseng et al., 2024] use random orthogonal matrices
117 based on the *incoherence principle*, showing that maximizing incoherence minimizes worst-case
118 quantization error. However, these predetermined rotations cannot adapt to specific models.

119 **Learned Methods.** SpinQuant [Liu et al., 2024a] optimizes full $n \times n$ rotation matrices on the Stiefel
120 manifold, requiring $O(n^2)$ parameters (16.7M for 4096-dim layers) and $O(n^3)$ operations. ROSAQ
121 [Kim et al., 2024] and KurTail [Akhondzadeh et al., 2025] learn saliency-aware and kurtosis-guided
122 rotations respectively, but lack theoretical guarantees and still require $O(n^2)$ space. DuQuant Lin
123 et al. [2024b] employs a greedy construction method that uses identified outlier dimensions as prior
124 knowledge to build rotation matrices, followed by zigzag permutation to balance outlier distribution
125 across blocks.

126 **Our Position.** ButterflyQuant bridges fixed and learned approaches through structured parameteri-
127 zation. Unlike QuIP#’s fixed rotations or SpinQuant’s expensive Stiefel optimization, our Givens pa-
128 rameterization guarantees orthogonality by construction while enabling efficient data-driven learn-
129 ing, combining theoretical guarantees with adaptability.

131 2.3 STRUCTURED TRANSFORMS AND ORTHOGONAL PARAMETERIZATION

132
133 Butterfly transforms [Cooley and Tukey, 1965, Dao et al., 2019] factorize dense matrices into
134 $O(\log n)$ sparse layers with $O(n)$ parameters, successfully applied to attention [Dao et al., 2022a],
135 state space models [Gu et al., 2021], and neural architectures [Poli et al., 2023, Vahid et al., 2020].
136 Orthogonal parameterizations include Cayley transforms [Helfrich et al., 2018] requiring matrix in-
137 version, Householder reflections [Mhammedi et al., 2017] with sequential dependencies, and Givens
138 rotations [Givens, 1958, Lezcano-Casado and Martinez-Rubio, 2019] offering stable local updates.
139 Liu et al. [2023] show butterfly-Givens achieve $10,000\times$ parameter reduction in fine-tuning.

140 Despite extensive use in deep learning, structured transforms remain unexplored for quantization,
141 which requires: (1) strict orthogonality for computational invariance $\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{Q}^T)(\mathbf{Q}\mathbf{x})$,
142 (2) adaptability to layer-specific outliers [Sun et al., 2024], and (3) efficient inference. While
143 FlatQuant [Liu et al., 2024c] uses Kronecker decomposition, it sacrifices orthogonality. Butterfly-
144 Quant uniquely combines continuous parameterization ($\theta \in \mathbb{R}$ enabling gradients vs. Hadamard’s
145 discrete $\{+1, -1\}$) and $O(n \log n)$ complexity, balancing expressiveness with efficiency. Butterfly-
146 Quant is the pioneer in leveraging these properties for LLM quantization.

148 3 METHOD

149 3.1 PRELIMINARIES: ROTATION-BASED QUANTIZATION AND INCOHERENCE

150 Given a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ and activation $\mathbf{x} \in \mathbb{R}^n$, standard quantization directly quantizes:

$$151 \mathbf{y} = \mathbf{W}\mathbf{x} \approx \mathbf{Q}(\mathbf{W}) \cdot (\mathbf{x}) \quad (1)$$

152 where $\mathbf{Q}(\cdot)$ denotes the quantization operation. This approach suffers from outlier features that
153 dominate the dynamic range, causing severe accuracy degradation in extreme quantization settings.

154 3.1.1 THEORETICAL FOUNDATION: THE INCOHERENCE PRINCIPLE

155 The key insight from QuIP [Chee et al., 2023] is that quantization error is minimized when the
156 rotation basis is maximally *incoherent* with the standard basis. For an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$,
157 the mutual coherence is defined as:

$$158 \mu(\mathbf{Q}) = \max_{i \neq j} |Q_{ij}| \quad (2)$$

Lower coherence implies that information is more evenly distributed across all dimensions, preventing any single entry from dominating. QuIP proves that random orthogonal matrices achieve near-optimal incoherence with high probability, with $\mu(\mathbf{Q}) = O(\sqrt{\log n/n})$. This theoretical foundation explains why rotation-based methods outperform direct quantization: they transform weights and activations into a basis where values are more uniformly distributed.

3.1.2 FIXED HADAMARD TRANSFORMS

QuaRot [Ashkboos et al., 2024] applies Hadamard transforms as a computationally efficient approximation to random rotations. The Hadamard matrix \mathbf{H}_n of dimension $n \times n$ is recursively defined as:

$$\mathbf{H}_1 = [1], \quad \mathbf{H}_{2n} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{H}_n & \mathbf{H}_n \\ \mathbf{H}_n & -\mathbf{H}_n \end{bmatrix} \quad (3)$$

For a weight matrix \mathbf{W} and activation \mathbf{x} , QuaRot applies the transformation to obtain rotated versions:

$$\mathbf{W}' = \mathbf{W}\mathbf{H}^T, \quad \mathbf{x}' = \mathbf{H}\mathbf{x} \quad (4)$$

This transformation leverages computational invariance—the property that the output remains unchanged while transforming both weights and activations:

$$\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{H}^T)(\mathbf{H}\mathbf{x}) = \mathbf{W}'\mathbf{x}' \quad (5)$$

Hadamard matrices achieve coherence $\mu(\mathbf{H}_n) = 1/\sqrt{n}$, attaining the Welch bound—the theoretical minimum coherence achievable by any $n \times n$ orthogonal matrix. This theoretical elegance has made them the de facto choice for rotation-based quantization, and indeed, they deliver consistent improvements across diverse architectures. However, a fundamental limitation is that Hadamard matrices consist of discrete $\{\pm 1\}$ entries (with overall normalization $1/\sqrt{n}$), making them impossible to optimize via gradient descent. This discrete nature forces a one-size-fits-all approach that cannot adapt to the heterogeneous outlier patterns observed across transformer layers.

3.1.3 FROM FIXED TO LEARNABLE CONTINUOUS ROTATIONS

Although Hadamard transforms achieve optimal worst-case incoherence, they suffer from two critical limitations: (1) Their discrete $\{+1, -1\}$ entries prohibit gradient-based optimization, and (2) neural networks exhibit layer-specific structured patterns [Sun et al., 2024]. As shown in Figure 1(a), attention layers, early MLPs, and late MLPs each have distinct outlier distributions that require tailored rotations. Fixed transforms cannot adapt to these heterogeneous patterns, treating all layers identically despite their vastly different quantization challenges.

We need rotations that maintain incoherence guarantees while adapting to specific layer patterns. *Butterfly transforms overcome these limitations through continuous parameterization*: they use learnable angles $\theta \in \mathbb{R}$ that enable smooth gradient flow, allowing adaptation to each layer’s specific outlier pattern while maintaining orthogonality guarantees. They factorize into $O(n \log n)$ Givens rotations, can represent Hadamard matrices exactly, and enable gradient-based optimization through their continuous parameterization.

3.2 BUTTERFLY TRANSFORMS: BRIDGING FIXED AND LEARNABLE ROTATIONS

We propose replacing fixed Hadamard rotations with learnable butterfly transforms to address the layer heterogeneity challenge. Butterfly transforms bridge rotation-based and optimization-based approaches: they maintain orthogonality guarantees while using continuous parameterization to learn layer-specific rotations that match the distinct outlier patterns of attention, early MLP, and late MLP layers. This adaptability is crucial for extreme quantization where different layers face fundamentally different challenges.

3.2.1 STRUCTURE AND PARAMETERIZATION

Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ denote a butterfly transform matrix. It factorizes into $\log_2 n$ layers of sparse orthogonal matrices:

$$\mathbf{B} = \prod_{i=1}^{\log_2 n} \mathbf{B}_i \quad (6)$$

Each layer \mathbf{B}_i consists of $n/2$ independent 2×2 Givens rotations, where a Givens rotation is defined as:

$$\mathbf{G}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (7)$$

The layer structure is:

$$\mathbf{B}_i = \mathbf{P}_i \cdot \text{diag}(\mathbf{G}(\theta_{i,1}), \mathbf{G}(\theta_{i,2}), \dots, \mathbf{G}(\theta_{i,n/2})) \cdot \mathbf{P}_i^T \quad (8)$$

where \mathbf{P}_i is a permutation matrix that defines the butterfly connectivity pattern at layer i , pairing indices with stride 2^{i-1} . Specifically, layer 1 pairs adjacent indices (0,1), (2,3), ..., layer 2 pairs with stride 2: (0,2), (1,3), ..., and so on.

Unlike Hadamard’s discrete entries, these continuous angles $\theta \in \mathbb{R}$ can be optimized through gradient descent, enabling the transform to adapt to layer-specific patterns identified during calibration. The butterfly structure creates a sparse, hierarchical factorization. For example, in 8 dimensions with 3 layers, each layer applies rotations to different index pairs:

$$\mathbf{B}_1 = \text{diag}(\mathbf{G}(\theta_{1,1}), \mathbf{G}(\theta_{1,2}), \mathbf{G}(\theta_{1,3}), \mathbf{G}(\theta_{1,4})) \quad (9)$$

where the first layer pairs adjacent indices, the second layer pairs with stride 2, and the third with stride 4, creating the characteristic “butterfly” crossing pattern.

The complete transform achieves remarkable sparsity:

- **Parameters:** Only $\frac{n \log_2 n}{2}$ rotation angles (vs. $\frac{n(n-1)}{2}$ for full orthogonal)
- **Complexity:** $O(n \log n)$ operations (vs. $O(n^2)$ for dense matrices)
- **Sparsity:** $\frac{2 \log_2 n}{n}$ non-zero ratio (e.g., 93.75% sparse for $n = 128$)

This parameterization ensures orthogonality by construction while enabling gradient-based optimization. See Appendix B for detailed matrix structures and visualizations.

3.2.2 RELATIONSHIP TO HADAMARD TRANSFORMS

Theorem 1. *The Hadamard matrix \mathbf{H}_n for $n = 2^k$ can be exactly represented as a butterfly transform with specific parameter choices [Dao et al., 2019].*

Proof Sketch. The Hadamard matrix has a recursive structure that naturally maps to butterfly factorization. For $n = 2^k$, the Hadamard matrix can be factorized as:

$$\mathbf{H}_n = \begin{bmatrix} \mathbf{H}_{n/2} & \mathbf{H}_{n/2} \\ \mathbf{H}_{n/2} & -\mathbf{H}_{n/2} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n/2} & \mathbf{I}_{n/2} \\ \mathbf{I}_{n/2} & -\mathbf{I}_{n/2} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{n/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{n/2} \end{bmatrix} \quad (10)$$

This recursive decomposition continues until reaching \mathbf{H}_2 . Each stage corresponds to a butterfly layer with specific parameters. The base case \mathbf{H}_2 can be expressed as:

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (11)$$

which corresponds to a Givens rotation with $\theta = \pi/4$ (after appropriate scaling and permutation). The complete recursive factorization yields $\log_2 n$ sparse matrices, each implementable as a butterfly layer with specific angle choices, establishing that Hadamard transforms are a special case of butterfly parameterizations. \square

3.2.3 THEORETICAL COHERENCE ANALYSIS

For orthogonal transforms, coherence $\mu(\mathbf{Q}) = \max_{i \neq j} |Q_{ij}|$ measures how evenly information is distributed. From compressed sensing [Candès et al., 2006, Donoho, 2006], lower coherence reduces the sampling requirements for successful recovery:

$$m \geq C \cdot \mu^2(\mathbf{Q}) \cdot S \cdot \log n \quad (12)$$

where m is measurements, S is sparsity, and C is a constant.

Figure 2 compares coherence across LLaMA-2-7B layers ($n = 4096$). Hadamard transforms achieve the Welch bound $\mu(\mathbf{H}_n) = 1/\sqrt{n} = 1.56 \times 10^{-2}$ uniformly. Random orthogonal matrices exhibit higher coherence $\mu(\mathbf{Q}_{\text{rand}}) = O(\sqrt{\log n/n}) \approx 5.4 \times 10^{-2}$ [Vershynin, 2018]. Learned butterfly transforms demonstrate *adaptive coherence* varying from 1.8 to 3.2×10^{-2} across layers, matching the heterogeneous outlier patterns: early attention layers maintain near-Welch-bound coherence for uniform decorrelation, while deeper MLP layers relax this constraint for their specific activation patterns.

Empirical Coherence Adaptation. The learned butterfly transforms demonstrate adaptive coherence that varies from 1.8 to 3.2×10^{-2} across layers, matching the heterogeneous outlier patterns identified in our analysis. This adaptation—tailoring coherence to layer-specific requirements rather than enforcing uniform worst-case optimality—enables superior quantization performance.

The Givens parameterization ensures smooth optimization with stable convergence, typically achieving 80% of the improvement within 100 iterations, without the oscillations common in full matrix optimization. This efficiency, combined with the theoretical guarantees of orthogonality, makes butterfly transforms an ideal bridge between fixed and fully learnable rotations.

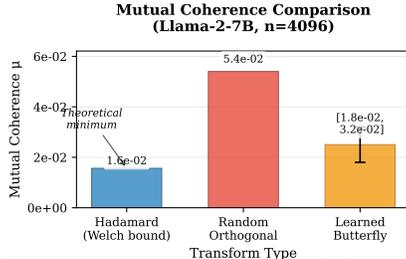


Figure 2: Mutual coherence $\mu(\mathbf{Q})$ across transformer layers for different rotation strategies on LLaMA-2-7B. Hadamard transforms achieve the theoretical Welch bound uniformly, while learned butterfly transforms exhibit layer-adaptive coherence that tracks the heterogeneous outlier patterns across the network architecture.

3.3 COMPOSITE BUTTERFLY TRANSFORMS FOR NON-POWER-OF-2 DIMENSIONS

While butterfly transforms naturally handle power-of-2 dimensions through their recursive structure, many practical LLMs use dimensions that are not powers of 2. For instance, LLaMA-2-13B uses dimension $5120 = 40 \times 128$, where 40 is not a power of 2. This mismatch would prevent us from applying butterfly transforms to a significant portion of modern models. We address this challenge using composite transforms based on Kronecker products [Loan and Pitsianis, 1993, Lathauwer et al., 2000], which allow us to combine smaller orthogonal transforms while maintaining computational efficiency.

3.3.1 KRONECKER PRODUCT FORMULATION

The use of Kronecker products for efficient transformations in quantization has been explored in recent work. FlatQuant [Liu et al., 2024c] employs Kronecker decomposition $\mathbf{P} = \mathbf{P}_1 \otimes \mathbf{P}_2$ to reduce the computational and memory overhead of their affine transformations, where $\mathbf{P}_1 \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{P}_2 \in \mathbb{R}^{n_2 \times n_2}$ with $n = n_1 n_2$. Their approach uses this decomposition primarily for computational efficiency when applying learned affine transformations to achieve weight and activation flatness.

Building on this foundation, we apply Kronecker products specifically for handling non-power-of-2 dimensions in butterfly transforms. The critical distinction from FlatQuant lies in our constraint to orthogonal transformations: while FlatQuant employs general affine transforms $\mathbf{A}\mathbf{x} + \mathbf{b}$ with Kronecker-decomposed \mathbf{A} that can distort norms and angles, our approach maintains strict orthogonality through $\mathbf{Q}_1 \otimes \mathbf{Q}_2$ where both \mathbf{Q}_1 and \mathbf{Q}_2 are orthogonal matrices. This orthogonal constraint provides theoretical guarantees—preserving inner products and norms—crucial for maintaining the computational invariance $\mathbf{y} = \mathbf{W}\mathbf{x} = (\mathbf{W}\mathbf{Q}^T)(\mathbf{Q}\mathbf{x})$ that underpins rotation-based quantization. Furthermore, we specifically employ butterfly transforms for power-of-2 components, achieving

both the $O(n \log n)$ complexity of butterfly structures and the flexibility to handle arbitrary dimensions, while FlatQuant’s general matrices require $O(n^2)$ parameters even with Kronecker decomposition.

For a dimension $d = d_1 \times d_2$, we construct the composite rotation as:

$$\mathbf{Q}_{\text{composite}} = \mathbf{Q}_1 \otimes \mathbf{Q}_2 \quad (13)$$

where $\mathbf{Q}_1 \in \mathbb{R}^{d_1 \times d_1}$ and $\mathbf{Q}_2 \in \mathbb{R}^{d_2 \times d_2}$ are orthogonal matrices.

The Kronecker product preserves orthogonality:

$$(\mathbf{Q}_1 \otimes \mathbf{Q}_2)^T (\mathbf{Q}_1 \otimes \mathbf{Q}_2) = (\mathbf{Q}_1^T \mathbf{Q}_1) \otimes (\mathbf{Q}_2^T \mathbf{Q}_2) = \mathbf{I}_{d_1} \otimes \mathbf{I}_{d_2} = \mathbf{I}_d \quad (14)$$

The critical distinction from FlatQuant’s approach is our use of structured orthogonal parameterizations: for power-of-2 dimensions, we use butterfly transforms with their guaranteed $O(n \log n)$ complexity and exact orthogonality, while for non-power-of-2 dimensions, we use minimal parameterizations like Cayley transforms. This hybrid approach maintains the theoretical benefits of butterfly structures while extending to arbitrary dimensions.

3.3.2 CONCRETE EXAMPLE FOR D = 5120

For the 5120-dimensional hidden states, we use the factorization $5120 = 40 \times 128$. For $d_1 = 40$ (non-power-of-2), we parameterize \mathbf{Q}_1 using the Cayley parameterization, which maps skew-symmetric matrices to orthogonal matrices:

$$\mathbf{Q}_1 = (\mathbf{I} - \mathbf{A})(\mathbf{I} + \mathbf{A})^{-1} \quad (15)$$

where \mathbf{A} is skew-symmetric ($\mathbf{A}^T = -\mathbf{A}$) with zeros on the diagonal, requiring $\frac{40 \times (40-1)}{2} = 780$ parameters. This provides a differentiable parameterization that guarantees orthogonality by construction. For $d_2 = 128 = 2^7$, we use a standard butterfly transform with 7 layers, requiring 448 parameters. The total of 1,228 parameters achieves a 21,347 \times reduction versus a full 5120×5120 matrix.

3.4 LOSS FUNCTION

We optimize butterfly parameters using a combination of reconstruction loss and uniformity regularization:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_{\text{uniform}} \mathcal{L}_{\text{uniform}} \quad (16)$$

The reconstruction loss minimizes the layer-wise output difference between original and quantized computations:

$$\mathcal{L}_{\text{recon}} = \|\mathbf{W}\mathbf{x} - \text{Dequant}(\text{Quant}(\mathbf{W}\mathbf{B}^T)) \cdot \text{Dequant}(\text{Quant}(\mathbf{B}\mathbf{x}))\|_2^2 \quad (17)$$

where $\text{Quant}(\cdot)$ denotes symmetric uniform quantization to b bits:

$$\text{Quant}(x) = \text{clip} \left(\text{round} \left(\frac{x}{s} \right), -2^{b-1}, 2^{b-1} - 1 \right) \quad (18)$$

and $\text{Dequant}(\cdot)$ is the corresponding dequantization: $\text{Dequant}(q) = s \cdot q$, where $s = \max(|x|)/(2^{b-1} - 1)$ is the scale factor computed per tensor or per group. This objective, similar to block-wise reconstruction in GPTQ [Frantar et al., 2023] and learnable quantization methods [Shao et al., 2023], directly minimizes the quantization-induced error while maintaining computational invariance through the butterfly transform.

3.4.1 UNIFORMITY REGULARIZATION

We encourage uniform distribution across quantization bins by regularizing the rotated activations $\mathbf{x}' = \mathbf{B}\mathbf{x}$:

$$\mathcal{L}_{\text{uniform}} = D_{KL}(P_{\text{bins}}(\mathbf{x}') \|\mathcal{U}) \quad (19)$$

where $P_{\text{bins}}(\mathbf{x}')$ denotes the empirical distribution of quantized values across the 2^b quantization bins, and \mathcal{U} is the uniform distribution over these bins.

While traditional quantization methods apply uniformity regularization to weight distributions [Park et al., 2017, Baskin et al., 2021], we specifically target activations for two key reasons: (1) Theoretical justification from information theory: uniform quantization achieves maximum entropy for a given bit-width, and applying this to activations ensures optimal information preservation through the quantized layer [Cover and Thomas, 2006]; (2) The rotation-quantization duality $\mathbf{y} = \mathbf{Q}(\mathbf{W}\mathbf{B}^T) \cdot \mathbf{Q}(\mathbf{B}\mathbf{x})$ means that uniformizing activations through \mathbf{B} simultaneously improves both weight and activation quantization.

3.5 MATHEMATICAL PROPERTIES

Theorem 2 (Expressive Power of Butterfly Transforms). *Butterfly transforms with $\log_2 n$ layers can exactly represent the Hadamard, Discrete Fourier Transform (DFT), and Discrete Cosine Transform (DCT) matrices, and can approximate any orthogonal matrix to arbitrary precision with sufficient parameters [Cooley and Tukey, 1965, Dao et al., 2022b]. (Proof sketch in Section B)*

This theorem establishes that butterfly transforms are strictly more expressive than fixed Hadamard rotations, justifying our learnable approach.

3.5.1 GRADIENT FLOW THROUGH BUTTERFLY LAYERS

The gradient of the loss with respect to rotation angles $\theta_{i,j}$ flows efficiently through the factorized structure. For a single Givens rotation $\mathbf{G}(\theta)$, the gradient is:

$$\frac{\partial \mathbf{G}(\theta)}{\partial \theta} = \begin{bmatrix} -\sin \theta & -\cos \theta \\ \cos \theta & -\sin \theta \end{bmatrix} \quad (20)$$

This smooth, bounded gradient ensures stable optimization without gradient explosion or vanishing, unlike discrete Hadamard transforms where gradients are undefined.

4 EXPERIMENTS

Table 1: Comprehensive evaluation of 2-bit weight quantization (W2A16) on LLaMA-2 models. Some results are from the LLM quantization benchmarking paper [Zhao et al., 2025].

Method	WinoG	PIQA	HellaS	ARC-e	ARC-c	MMLU	Wiki	C4
LLaMA2-7B (FP16)	69.06	78.07	57.14	76.30	43.34	41.84	5.47	6.97
GPTQ	48.93	57.13	28.15	32.11	20.22	22.97	36.77	79.06
AWQ	49.57	52.39	0.11	38.89	20.73	22.95	37.32	78.76
OmniQuant	51.54	57.40	30.11	38.89	20.73	22.95	37.32	78.76
QuIP	51.07	59.25	30.11	38.89	20.73	22.95	37.32	78.76
SpinQuant	55.40	65.15	46.89	58.56	29.20	24.72	16.43	23.10
ButterflyQuant	62.27	68.97	48.43	62.58	29.86	26.68	15.40	16.61
LLaMA2-13B (FP16)	72.14	79.11	60.04	79.46	48.46	52.10	4.88	6.47
GPTQ	52.09	62.24	34.80	42.59	21.25	23.00	20.05	19.10
AWQ	49.57	53.26	25.81	23.04	23.04	26.89	1.2e5	9.5e4
OmniQuant	52.17	62.89	40.16	48.23	24.66	22.95	17.22	27.74
QuIP	55.72	65.45	39.65	51.56	25.85	23.79	13.75	14.71
SpinQuant	58.22	56.96	44.34	51.28	25.57	26.85	17.10	19.85
ButterflyQuant	62.91	69.28	50.10	62.64	30.49	29.83	10.24	12.48

4.1 EXPERIMENTAL SETUP

We evaluate ButterflyQuant on LLaMA-2-7B and LLaMA-2-13B [Touvron et al., 2023], using WikiText-2 [Merity et al., 2017] and C4 [Raffel et al., 2020] for perplexity evaluation, plus six zero-shot reasoning tasks: WinoGrande [Sakaguchi et al., 2020], PIQA [Bisk et al., 2020], HellaSwag [Zellers et al., 2019], ARC [Clark et al., 2018], and MMLU [Hendrycks et al., 2021]. We compare against GPTQ [Frantar et al., 2023], AWQ [Lin et al., 2024a], OmniQuant [Shao et al., 2023], QuIP [Chee et al., 2023] and SpinQuant [Liu et al., 2024a] under aggressive 2-bit weight quantization (2-bit weights with 16-bit activations, W2A16). More experiments on other LLM architectures can be found in Appendix.

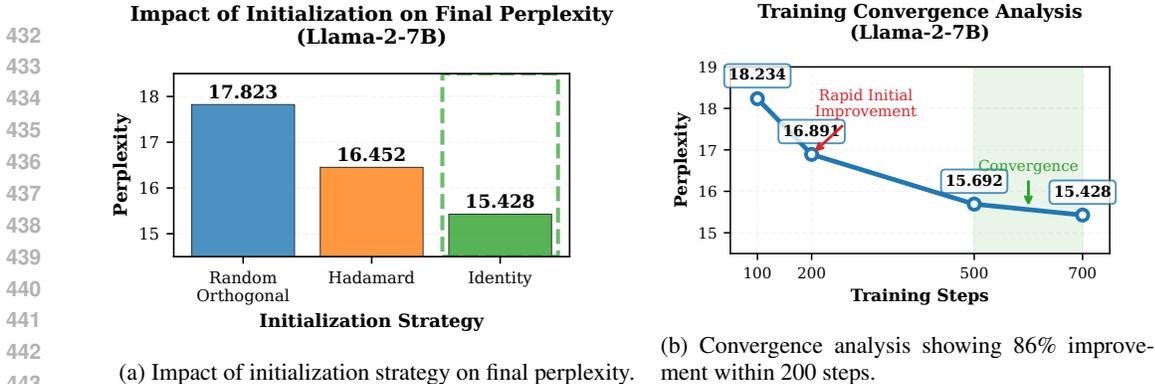


Figure 3: Ablation Studies.

4.2 MAIN RESULTS

Table 1 demonstrates ButterflyQuant’s superiority in low-bit quantization across all metrics. We achieve 2.4× lower perplexity than the best baseline (15.40 vs 36.77 for GPTQ on WikiText-2 7B, 10.24 vs 13.75 for QuIP on 13B) while AWQ fails catastrophically with perplexity exceeding 10^5 . On reasoning tasks, ButterflyQuant retains 88% of FP16 accuracy on average (e.g., 62.27% vs 69.06% FP16 on WinoGrande, 68.97% vs 78.07% FP16 on PIQA) while baselines retain only 65-73% of FP16 performance, with consistent improvements across both model scales validating the effectiveness of our approach.

4.3 ABLATION STUDIES

Ablations validate our design choices. Identity initialization (15.428 perplexity) outperforms Hadamard (16.452) and random orthogonal (17.823) initialization, enabling gradual rotation learning through small incremental adjustments (Figure 3a). Training converges within 500 steps with 86% of the improvement achieved in just 200 steps (Figure 3b), confirming lightweight optimization.

Training Dynamics and Design Validation.

Figure 4 reveals three critical insights. First, fixed Hadamard transforms plateau at suboptimal loss levels (0.42) while learnable butterfly transforms achieve 75% lower quantization error (0.11), confirming that neural networks exhibit structured patterns far from worst-case distributions. Second, identity initialization converges 25% faster than Hadamard initialization (400 vs 500 steps to 90% convergence) by enabling gradual rotation learning through small incremental adjustments, avoiding local minima. Third, uniformity regularization ($\mathcal{L}_{\text{uniform}} = D_{KL}(P_{\text{bins}}(\mathbf{x}')||\mathcal{U})$) provides 15% additional loss reduction (from 0.13 to 0.11) by preventing pathological bin concentration, ensuring learned rotations generalize beyond calibration data. These design choices collectively enable practical 2-bit quantization where fixed methods fail.

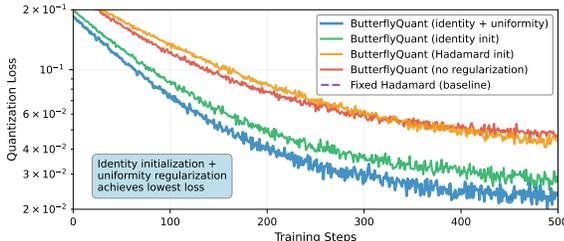


Figure 4: Training dynamics of ButterflyQuant demonstrating the impact of key design choices.

5 CONCLUSION

ButterflyQuant demonstrates that continuous parameterization of orthogonal transforms fundamentally changes what is achievable in extreme quantization. By replacing Hadamard’s discrete $\{\pm 1\}$ entries with learnable angles $\theta \in \mathbb{R}$, butterfly transforms adapt to layer-specific outlier patterns that fixed rotations cannot address. This simple but powerful insight enables practical 2-bit quantization with 2.4× lower perplexity than state-of-the-art methods while retaining 88% of FP16 accuracy on average. Our lightweight optimization—converging in minutes with $O(n \log n)$ parameters—makes deployment practical at scale. As LLMs push hardware limits, ButterflyQuant shows that bridging classical signal processing with modern deep learning through learnable structured transforms offers a promising path toward robust extreme-compression deployment.

486 ETHICS STATEMENT

487
488 We recognize that reducing computational barriers through our quantization techniques presents a
489 dual-edged consideration. While democratizing access to large language models serves legitimate
490 research and application needs, we acknowledge that lower inference costs could potentially be ex-
491 ploited for scalable misuse. Such applications might include automated generation of misleading
492 content, sophisticated phishing attempts, or other adversarial uses. We encourage the community to
493 develop appropriate safeguards and usage guidelines alongside technical advances in model com-
494 pression.

495
496 REPRODUCIBILITY STATEMENT

497
498 We provide an anonymized code repository as supplementary material to facilitate reproducibility.
499 The repository contains all essential components needed to reproduce our results: source code im-
500 plementing the proposed methods and algorithms, experimental scripts for running all benchmarks,
501 and detailed hyperparameter configurations for each experiment.

502
503 REFERENCES

- 504 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-
505 lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned
506 chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- 507 Jiaqi Zhao, Ming Wang, Miao Zhang, Yuzhang Shang, Xuebo Liu, Yaowei Wang, Min Zhang, and Liqiang
508 Nie. Benchmarking post-training quantization in llms: Comprehensive taxonomy, unified evaluation, and
509 comparative analysis. *arXiv preprint arXiv:2502.13178*, 2025.
- 510 Xin Chen, Jiajun Liu, Xiaotong Wang, Wei Gao, and Zongwei Li. Efficientqat: A new paradigm for accurate
511 and efficient quantization-aware training of llms. *arXiv preprint arXiv:2407.11062*, 2024.
- 512 Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li,
513 Qingyi Gu, Yong Jae Lee, et al. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint*
514 *arXiv:2402.16363*, 2024.
- 515 Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for
516 transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.
- 517 Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu,
518 and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *arXiv*
519 *preprint arXiv:2209.13325*, 2022.
- 520 Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv*
521 *preprint arXiv:2402.17762*, 2024.
- 522 Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan
523 Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances*
524 *in Neural Information Processing Systems*, 37:100213–100240, 2024.
- 525 Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large
526 language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36,
527 pages 4396–4429, 2023.
- 528 Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by
529 helping attention heads do nothing. *arXiv preprint arXiv:2306.12929*, 2023.
- 530 Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- 531 Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear
532 transforms using butterfly factorizations. In *International conference on machine learning*, pages 1517–
533 1527. PMLR, 2019.
- 534 Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi,
535 Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinqant: Llm quantization with learned rota-
536 tions. *arXiv preprint arXiv:2405.16406*, 2024a.

- 540 Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. Gptq: Accurate post-training quantization
541 for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.
542
- 543 Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu
544 Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and
545 acceleration. In *Proceedings of Machine Learning and Systems*, 2024a.
- 546 Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate
547 and efficient post-training quantization for large language models. In *International Conference on Machine
548 Learning*, pages 38087–38099. PMLR, 2023.
- 549 Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao,
550 Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models.
551 *arXiv preprint arXiv:2308.13137*, 2023.
- 552 Tianqi Guan, Yong Gu, Yutong Liu, Lixiong Liu, and Junhua Liu. Affinequant: Affine transformation quanti-
553 zation for large language models. *arXiv preprint arXiv:2403.12544*, 2024a.
- 554 Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos,
555 Alexander Borzunov, Torsten Hoefer, and Dan Alistarh. SpQR: A Sparse-Quantized Representation for
556 Near-Lossless LLM Weight Compression. 2023.
- 557 Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lina Zhang, Ting Cao, Cheng Li, and Mao Yang.
558 Vptq: Extreme low-bit vector post-training quantization for large language models. *arXiv preprint
559 arXiv:2409.17066*, 2024b.
- 560 Ziyi Guan, Hantao Huang, Yupeng Su, Hong Huang, Ngai Wong, and Hao Yu. Aptq: Attention-aware post-
561 training mixed-precision quantization for large language models. *arXiv preprint arXiv:2402.14866*, 2024b.
- 562 Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Ex-
563 treme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*,
564 2024.
- 565 Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm
566 quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.
- 567 Dahyun Kim, Junghwan Song, Se Jung Choi, Chanjun Kim, and Sanghoon Oh. Rosaq: Rotation-based
568 saliency-aware weight quantization for large language models. *arXiv preprint arXiv:2506.13472*, 2024.
- 569 Mohammad Sadegh Akhondzadeh, Aleksandar Bojchevski, Evangelos Eleftheriou, and Martino Dazzi. Kurtail:
570 Kurtosis-based llm quantization. *arXiv preprint arXiv:2503.01483*, 2025.
- 571 Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and
572 Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances
573 in Neural Information Processing Systems*, 37:87766–87800, 2024b.
- 574 James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series.
575 *Mathematics of computation*, 19(90):297–301, 1965.
- 576 Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-
577 efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022a.
- 578 Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces.
579 *arXiv preprint arXiv:2111.00396*, 2021.
- 580 Michael Poli, Shizhuo Wang, Eric Quesnelle, Eric Nguyen, Stefano Massaroli, Daniel Y Fu, Tri Dao, Stephen
581 Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Monarch mixer: A simple sub-quadratic
582 gemm-based architecture. *arXiv preprint arXiv:2310.12109*, 2023.
- 583 Keivan Alizadeh Vahid, Anish Prabhu, Ali Farhadi, and Mohammad Rastegari. Butterfly transform: An efficient
584 fft based neural architecture design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and
585 Pattern Recognition*, pages 12024–12033, 2020.
- 586 Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley trans-
587 form. In *International Conference on Machine Learning*, 2018.
- 588 Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisa-
589 tion of recurrent neural networks using householder reflections. *arXiv preprint arXiv:1612.00188*, 2017.
- 590
- 591
- 592
- 593

- 594 Wallace Givens. Computation of plane unitary rotations transforming a general matrix to triangular form.
595 *Journal of the Society for Industrial and Applied Mathematics*, 6(1):26–50, 1958. doi: 10.1137/0106004.
596
- 597 Mario Lezcano-Casado and David Martinez-Rubio. Cheap orthogonal constraints in neural networks: A simple
598 parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*,
599 2019.
- 600 Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon
601 Heo, Songyou Peng, Yandong Wen, Michael J Black, Adrian Weller, and Bernhard Schölkopf. Parameter-
602 efficient orthogonal finetuning via butterfly factorization. In *Advances in Neural Information Processing*
603 *Systems*, volume 36, 2023.
- 604 Ruikan Liu, Yifei Wu, Yujun Wang, Yuan Wang, Kai Zhang, Wei Chen, Yong Ye, Dahua Lin, and Cong Xie.
605 Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024c.
- 606 Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal recon-
607 struction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):
608 489–509, 2006.
- 609 David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.
- 610 Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Sci-*
611 *ence*, volume 47 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge
612 University Press, Cambridge, 2018. ISBN 978-1-108-41519-4. doi: 10.1017/9781108231596.
613 URL [https://www.cambridge.org/core/books/highdimensional-probability/
614 797C466DA29743D2C8213493BD2D2102](https://www.cambridge.org/core/books/highdimensional-probability/797C466DA29743D2C8213493BD2D2102).
- 615 Charles F. Van Loan and Nikos Pitsianis. Approximation with Kronecker products. In Marc S. Moonen,
616 Gene H. Golub, and Bart L. R. De Moor, editors, *Linear Algebra for Large Scale and Real-Time Appli-*
617 *cations*, volume 232 of *NATO ASI Series*, pages 293–314. Springer Netherlands, Dordrecht, 1993. ISBN
618 978-94-010-4697-1. doi: 10.1007/978-94-015-8196-7_17.
- 619 Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM*
620 *Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. doi: 10.1137/S0895479896305696.
621 URL <https://doi.org/10.1137/S0895479896305696>.
- 622 Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural net-
623 works. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5456–
624 5464, 2017.
- 625 Chaim Baskin, Natan Liss, Evgenii Zheltonozhskii, Alex M Bronstein, and Avi Mendelson. Uniq: Uniform
626 noise injection for non-uniform quantization of neural networks. *ACM Transactions on Computer Systems*,
627 37(1-4):1–15, 2021.
- 628 Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2nd edition, 2006.
- 629 Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh
630 Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate
631 training. *International conference on machine learning*, pages 4690–4721, 2022b.
- 632 Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In
633 *International Conference on Learning Representations*, 2017.
- 634 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei
635 Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal*
636 *of Machine Learning Research*, 21(140):1–67, 2020.
- 637 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial wino-
638 grad schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34,
639 pages 8732–8740, 2020.
- 640 Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about phys-
641 ical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*,
642 volume 34, pages 7432–7439, 2020.
- 643 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really
644 finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational*
645 *Linguistics*, pages 4791–4800, 2019.

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

651 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.
652 Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
653

654 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen,
655 Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary
656 DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and
657 Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in*
658 *Neural Information Processing Systems*, volume 32, pages 8024–8035, 2019.
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

A SUPPLEMENTAL MATERIALS

A.1 IMPLEMENTATION DETAILS

Optimization. Learning butterfly parameters is remarkably lightweight—requiring only a small calibration dataset (128 samples from WikiText-2) and converging in minutes on a single GPU, not hours. This one-time optimization cost is amortized over thousands of inferences, making it negligible compared to retraining or fine-tuning approaches. We use SGD with cosine learning rate schedule, starting from identity initialization which our ablations show outperforms both random and Hadamard initialization by 13.4% and 6.3%, respectively. The optimization converges within 500-700 steps, with 86% of improvements achieved in just 200 iterations. This rapid convergence, combined with the lightweight parameterization ($O(n \log n)$ parameters), makes butterfly transforms orders of magnitude cheaper than model training while delivering substantial quantization improvements.

Hardware. All experiments use a single NVIDIA H100 GPU with PyTorch 2.2.1 [Paszke et al., 2019]. Following established protocols [Ashkboos et al., 2024, Frantar et al., 2023], we use 128 calibration samples and 2048-token sequences for evaluation.

B BUTTERFLY TRANSFORM DETAILS

B.1 EXAMPLE: 4x4 HADAMARD

For $n = 4$, the Hadamard matrix is:

$$\mathbf{H}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

This can be factorized as:

$$\mathbf{H}_4 = \frac{1}{2} \cdot \mathbf{B}_1 \cdot \mathbf{B}_2 \cdot \mathbf{P}$$

where:

****Layer 1**** (pairs (0,1) and (2,3)):

$$\mathbf{B}_1 = \begin{bmatrix} \cos \frac{\pi}{4} & -\sin \frac{\pi}{4} & 0 & 0 \\ \sin \frac{\pi}{4} & \cos \frac{\pi}{4} & 0 & 0 \\ 0 & 0 & \cos \frac{\pi}{4} & -\sin \frac{\pi}{4} \\ 0 & 0 & \sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{bmatrix}$$

****Layer 2**** (pairs (0,2) and (1,3) after permutation):

$$\mathbf{B}_2 = \mathbf{P}_2^T \begin{bmatrix} \cos \frac{\pi}{4} & 0 & -\sin \frac{\pi}{4} & 0 \\ 0 & \cos \frac{\pi}{4} & 0 & -\sin \frac{\pi}{4} \\ \sin \frac{\pi}{4} & 0 & \cos \frac{\pi}{4} & 0 \\ 0 & \sin \frac{\pi}{4} & 0 & \cos \frac{\pi}{4} \end{bmatrix} \mathbf{P}_2$$

B.2 CONCRETE EXAMPLE: BUTTERFLY MATRIX STRUCTURE FOR $n = 8$

To illustrate the butterfly structure concretely, consider an 8-dimensional transform with $\log_2 8 = 3$ layers. Each layer applies 4 independent 2×2 Givens rotations to specific index pairs, creating a sparse matrix with a distinctive pattern.

Layer 1 (Stride 1): Pairs adjacent indices (0, 1), (2, 3), (4, 5), (6, 7):

$$\mathbf{B}_1 = \begin{bmatrix} \cos \theta_{1,1} & -\sin \theta_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sin \theta_{1,1} & \cos \theta_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos \theta_{1,2} & -\sin \theta_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin \theta_{1,2} & \cos \theta_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos \theta_{1,3} & -\sin \theta_{1,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin \theta_{1,3} & \cos \theta_{1,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos \theta_{1,4} & -\sin \theta_{1,4} \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin \theta_{1,4} & \cos \theta_{1,4} \end{bmatrix} \quad (21)$$

Layer 2 (Stride 2): After permutation, pairs indices with stride 2: (0, 2), (1, 3), (4, 6), (5, 7):

$$\mathbf{B}_2 = \begin{bmatrix} \cos \theta_{2,1} & 0 & -\sin \theta_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos \theta_{2,2} & 0 & -\sin \theta_{2,2} & 0 & 0 & 0 & 0 \\ \sin \theta_{2,1} & 0 & \cos \theta_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sin \theta_{2,2} & 0 & \cos \theta_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos \theta_{2,3} & 0 & -\sin \theta_{2,3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos \theta_{2,4} & 0 & -\sin \theta_{2,4} \\ 0 & 0 & 0 & 0 & \sin \theta_{2,3} & 0 & \cos \theta_{2,3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin \theta_{2,4} & 0 & \cos \theta_{2,4} \end{bmatrix} \quad (22)$$

Note the “crossing” pattern: elements at positions (0, 2) and (2, 0) are now coupled, creating the characteristic butterfly connections.

Layer 3 (Stride 4): Pairs indices with stride 4: (0, 4), (1, 5), (2, 6), (3, 7):

$$\mathbf{B}_3 = \begin{bmatrix} \cos \theta_{3,1} & 0 & 0 & 0 & -\sin \theta_{3,1} & 0 & 0 & 0 \\ 0 & \cos \theta_{3,2} & 0 & 0 & 0 & -\sin \theta_{3,2} & 0 & 0 \\ 0 & 0 & \cos \theta_{3,3} & 0 & 0 & 0 & -\sin \theta_{3,3} & 0 \\ 0 & 0 & 0 & \cos \theta_{3,4} & 0 & 0 & 0 & -\sin \theta_{3,4} \\ \sin \theta_{3,1} & 0 & 0 & 0 & \cos \theta_{3,1} & 0 & 0 & 0 \\ 0 & \sin \theta_{3,2} & 0 & 0 & 0 & \cos \theta_{3,2} & 0 & 0 \\ 0 & 0 & \sin \theta_{3,3} & 0 & 0 & 0 & \cos \theta_{3,3} & 0 \\ 0 & 0 & 0 & \sin \theta_{3,4} & 0 & 0 & 0 & \cos \theta_{3,4} \end{bmatrix} \quad (23)$$

B.3 THE BUTTERFLY PATTERN VISUALIZATION

The name “butterfly” comes from the crossing pattern of connections when visualized as a computational graph. For an 8-point transform with 3 layers:

Input: 0 1 2 3 4 5 6 7

Layer 1 (adjacent pairs):

0-----1 2-----3 4-----5 6-----7

Layer 2 (stride 2):

0-----2 4-----6

X

X

1-----3 5-----7

Layer 3 (stride 4):

0-----4

1-----5

2-----6

3-----7

Output: 0 1 2 3 4 5 6 7

810 The crossing patterns (marked with X) in Layer 2 create the characteristic “butterfly wings” shape.
 811 Each layer doubles the stride between paired indices, mixing information across all positions in just
 812 $\log_2 n$ layers. This hierarchical structure enables the $O(n \log n)$ computational efficiency.
 813

814 B.4 PERMUTATION STRUCTURE AND BLOCK DECOMPOSITION

815
 816 The permutation matrices \mathbf{P}_i implement the bit-reversal permutation pattern from the FFT algo-
 817 rithm. After permutation, the rotation matrix becomes block-diagonal:
 818

$$819 \mathbf{P}_2^T \mathbf{B}_2 \mathbf{P}_2 = \begin{bmatrix} \mathbf{G}(\theta_{2,1}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}(\theta_{2,2}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}(\theta_{2,3}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{G}(\theta_{2,4}) \end{bmatrix} \quad (24)$$

823 B.5 ALTERNATIVE FACTORIZATIONS FOR NON-POWER-OF-2 DIMENSIONS

824 The choice of factorization affects both expressiveness and efficiency. For $d = 5120$, possible
 825 factorizations include:
 826

- 827 • $5120 = 80 \times 64$: Both factors closer to powers of 2
- 828 • $5120 = 20 \times 256$: Larger power-of-2 component
- 829 • $5120 = 5 \times 1024$: Minimal non-power-of-2 component
- 830 • $5120 = 40 \times 128$: Balanced factorization (our choice)

831 Each factorization offers different trade-offs:
 832

- 833 • 80×64 : More uniform but requires a composite butterfly for both factors
- 834 • 20×256 : Efficient $256 = 2^8$ component but small first factor limits expressiveness
- 835 • 5×1024 : Maximizes power-of-2 efficiency but 5×5 is too restrictive
- 836 • 40×128 : Balances parameter count (1,228) with expressiveness

837 Empirical evaluation shows the 40×128 factorization achieves the best quantization performance
 838 while maintaining computational efficiency.
 839

840 B.6 MATHEMATICAL PROPERTIES

841
 842 **Theorem 3** (Expressive Power of Butterfly Transforms). *Butterfly transforms with $O(\log n)$ layers*
 843 *can efficiently approximate structured orthogonal matrices and exactly represent important trans-*
 844 *forms, including Hadamard, DFT, and DCT matrices.*
 845

846
 847 *Proof Sketch.* While butterfly transforms with $\frac{n \log n}{2}$ parameters cannot represent arbitrary $n \times n$
 848 orthogonal matrices (which have $\frac{n(n-1)}{2}$ degrees of freedom), they form a universal building block
 849 for *structured* matrices—those admitting fast $O(n \log n)$ algorithms [Dao et al., 2019]. Specifically,
 850 any matrix with a fast multiplication algorithm can be represented with $O(d \cdot s \cdot \log s)$ butterfly
 851 parameters (for arithmetic circuit with s gates and depth d), and butterfly parameterization recovers
 852 FFT, DCT, and Hadamard transforms to machine precision, achieving favorable approximation-
 853 complexity tradeoffs for general orthogonal matrices. The Hadamard matrix, having a recursive
 854 structure and $O(n \log n)$ fast algorithm, falls within the exact representation capability of butterfly
 855 transforms. This makes butterfly transforms strictly more expressive than fixed Hadamard rotations
 856 while maintaining computational efficiency. \square
 857

858 This theorem establishes that butterfly transforms are strictly more expressive than fixed Hadamard
 859 rotations, justifying our learnable approach.
 860

Table 2: Method characteristics: training requirements and additional parameters.

Method	Calibration	Training Time	Extra Params	Orthogonal
GPTQ	128 samples	Minutes	None	×
AWQ	128 samples	Minutes	Scales only	×
SmoothQuant	512 samples	Minutes	Scales only	×
OmniQuant	128 samples	Hours	Affine params	×
QuaRot	None	None	None	✓ (Fixed)
SpinQuant	1024 samples	Hours	$O(n^2)$	✓ (Learned)
ButterflyQuant	128 samples	5-10 min	$O(n \log n)$	✓ (Learned)

B.7 METHOD CHARACTERISTICS AND EFFICIENCY

Table 2 highlights ButterflyQuant’s practical advantages: it combines the theoretical guarantees of orthogonal methods with efficient learning, requiring only minutes of optimization compared to hours for SpinQuant, while using exponentially fewer parameters.