WATERKRON AND FLIPFLOP HESSIAN: INFORMATION-THEORETICALLY GROUNDED QUANTIZATION WITH KRONECKER-FACTORED HESSIANS

Anonymous authors
Paper under double-blind review

ABSTRACT

How should a Kronecker-factored Hessian approximation be chosen for post-training quantization? We address this question through *WaterKron*, which combines two-sided GPTQ with row- and column-dependent waterfilling scales and entropy coding. We derive its high-rate distortion with respect to the full Hessian using an explicit Kronecker-Hessian mismatch factor Φ. This factor quantifies the asymptotic distortion penalty due to the Kronecker Hessian approximation and provides a criterion for selecting the factors optimally. Minimizing Φ leads to a Gaussian covariance-fitting problem with classical "flip-flop" updates. We thus give a rate-distortion justification for using the resulting *FlipFlop Hessian* in quantization. We evaluate it empirically, finding that the FlipFlop Hessian consistently improves KL divergence and perplexity over input-only, marginal, and Frobenius-based Hessian choices.

1 INTRODUCTION

Post-training quantization of neural networks is the process of taking a fully trained neural network and converting its high-precision weights into a low-precision format, with the goal of retaining network accuracy while reducing memory requirements. Usually, each linear module W is quantized separately. Hence the goal is to find a low-precision approximation V of W.

There exist different classes of algorithms for finding such a V, which can also be combined. Some methods focus on making the entries of W incoherent and thus easier to quantize, for example by utilizing random or learned rotations (Chee et al., 2023; Ashkboos et al., 2024; Liu et al., 2025). Other methods change the base quantizer that is applied to a single entry or a small group of entries, for example by using codebooks or lattices instead of simple rounding (Tseng et al., 2024a; Savkin et al., 2025). A third class of methods does "adaptive rounding" by conditioning the quantization of later entries within W such that they correct the errors introduced by quantizing earlier entries.

Our work falls within the realm of this third class of methods, more concretely within its largest subclass, which might be labeled "error propagation under a quadratic loss". A prominent example within this line of research is the GPTQ algorithm by Frantar et al. (2023). In order to measure the distortion when W is replaced by V, the algorithm uses a quadratic loss. However, instead of just looking at $\|V - W\|_F$, for which simple "round to nearest" is the optimal quantization algorithm, it takes into account the input $x \in \mathbb{R}^n$ of the linear module W, and measures the distortion as $\mathbb{E}\|Vx - Wx\|_2^2$. In practice, the expectation is taken over a calibration dataset. This algorithm has been analyzed and extended from different directions (e.g., Chee et al., 2023; Tseng et al., 2024a;b; Zhang et al., 2026; Tseng et al., 2026; Birnick, 2026; Chen et al., 2026b; Zhang et al., 2025; Ordentlich & Polyanskiy, 2026; Lifar et al., 2026), but central to this paper are:

1. The extension to a **two-sided geometry** (Kim et al., 2025; Tseng et al., 2026; Birnick & Saab, 2026). In standard GPTQ, the rows of $W \in \mathbb{R}^{m\times n}$ are quantized independently. Formally, when we vectorize the matrices W, V into $w = \mathrm{vec}(W) \in \mathbb{R}^{mn}, v = \mathrm{vec}(V)$, then the Hessian of the loss $\mathbb{E}\|(V - W)x\|_2^2/2$ with respect to v is $\mathbb{E}[xx^T] \otimes I_m$. While the $\mathbb{E}[xx^T]$ factor corresponds to the input geometry that GPTQ exploits, the identity factor I_m means that all output features are treated independently and identically. There is a natural

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

extension of GPTQ that can deal with arbitrary Kronecker-factored Hessians $A \otimes B$ and reduces to standard GPTQ if $B = I_m$. This variant of the algorithm effectively minimizes the loss $\|(B^{1/2})^T(V - W)A^{1/2}\|_F^2$; in particular, the factor B in the Hessian also informs the algorithm of the output geometry of the linear module W. For the exact choice of the Hessian factors A and B, multiple options have been explored in the literature (see Section 5 and Section 6).

2. The **information-theoretic** view and waterfilling formula (Lifar et al., 2026; Ordentlich & Polyanskiy, 2026). Standard GPTQ allocates the same number of bits for every entry of V. An alternative is to use *entropy coding* (Chen et al., 2026b; Lifar et al., 2026) for the quantized matrix. In particular, this allows us to ask about precision/bit/rate allocation: We have the freedom to store some entries in "higher precision" (usually costing more bits), and other entries in "lower precision" (using fewer bits). Concretely, "precision" corresponds to the scaling factors of the quantization grid. For example, Huffman-GPTQ (Chen et al., 2026b), which does use Huffman coding and thus encodes some entries with more bits and others with fewer, still distributes the precision/rate uniformly across all entries. If the variance $\mathrm{Var}(x_j)$ of the jth input channel is known, a more effective way to distribute precision would be to assign to the jth column of W a scale proportional to $1/\sqrt{\mathrm{Var}(x_j)}$. However, Ordentlich & Polyanskiy (2026); Lifar et al. (2026) notice that even that is suboptimal: In GPTQ, we can exploit linear correlation between different input channels, and what matters is only the residual variance that remains after accounting for this correlation. So the scale for the jth column should be chosen proportional to $1/\sqrt{\min_{e_{j+1},\dots,e_n} \mathrm{Var}(x_j - e_{j+1}x_{j+1} - \cdots - e_n x_n)}$, and this is essentially optimal.

In this paper, we apply the information-theoretic lens to the two-sided algorithm. First, this yields *WaterKron*, an extension of the existing waterfilling allocation, WaterSIC, to the Kronecker-factored geometry. Then we characterize its distortion in the full Hessian geometry, identify the penalty caused by the Kronecker approximation, and use it to find the optimal choice of Kronecker approximation. Concretely, we make the following contributions:

1. **WaterKron: two-sided precision allocation.** We extend the high-rate waterfilling results to two-sided GPTQ, deriving optimal row and column scales for the analyzed scalar-grid alphabet. Under the Gaussian and entropy-coding assumptions, this retains the approximately 0.255-bit asymptotic gap to the information-theoretic limit for the Kronecker distortion (Corollary 3.2). We demonstrate empirically that this outperforms allocating the waterfilling precision using only the input-sided Hessian (WaterSIC).
2. **Quantifying the cost of Hessian mismatch.** We characterize WaterKron's high-rate distortion in the *full* Hessian geometry under the same assumptions. We introduce a mismatch factor Φ that quantifies the multiplicative distortion penalty caused by the Kronecker approximation, providing a criterion for choosing its factors (Theorem 4.1).
3. **The FlipFlop Hessian.** Minimizing the mismatch factor, we recover the flip-flop algorithm from Gaussian covariance fitting. We propose the resulting *FlipFlop Hessian approximation*, which is justified by its distortion-minimizing objective and, to the best of our knowledge, has not been explored in this line of quantization research (Section 5). We demonstrate empirically that it outperforms previous choices of Hessian approximations from the quantization literature.

Our results extend the high-rate waterfilling theory of Ordentlich & Polyanskiy (2026); Lifar et al. (2026) to Kronecker-factored quadratic losses. They recover (and improve), as a special case, the row- and column-dependent grid allocation of SoftWater (Cavalcanti & Wilson, 2026), which is designed for the softmax module with a diagonal approximation of the output curvature B.

The rest of the paper is structured as follows. In Section 2 we recall the setting and previous work: standard GPTQ, its two-sided extension, and the waterfilling precision allocation. In Section 3 we first derive the optimal precision allocation for the two-sided algorithm. In Section 4 we take an information-theoretic viewpoint on the optimal choice of the Kronecker-factored Hessian approximation. We introduce the Kronecker-Hessian mismatch factor Φ and provide a variant of the rate-distortion result that takes the full Hessian into account. In Section 5, informed by the mismatch theory, we derive the ideal Kronecker approximation, proposing the FlipFlop Hessian. Lastly, we provide empirical evidence for the success of our methods in Section 6.

2 SETTING AND BACKGROUND

In this section, we first describe the alphabet that is used for the remainder of the paper. Then we recall the standard GPTQ algorithm, its two-sided extension, and the waterfilling precision allocation.

Notation. For an m by n matrix M, $\mathrm{vec}(M) \in \mathbb{R}^{mn}$ is its column-major vectorization. Assuming $m = n$, we abbreviate $|M| := \det M$. If M is additionally positive definite, $\mathrm{Cholesky}(M)$ denotes the unique lower-triangular matrix L with positive diagonal entries $L_{i,i} > 0$ that satisfies $LL^T = M$. Similarly, $\mathrm{RevCholesky}(M)$ denotes the unique such L that satisfies $L^T L = M$. $M^{1/2}$ denotes an arbitrary matrix S satisfying $SS^T = M$, the most important choice for this paper being $S = \mathrm{RevCholesky}(M)^T$. All PSD matrices are assumed to be invertible; in practice, a small damping κI can be added to ensure this. Lastly, $\{0,1\}^* = \bigcup_{n=0}^{\infty} \{0,1\}^n$ is the set of finite binary strings.

2.1 ALPHABET

Recall that the goal of weight-only post-training quantization is to approximate $W \in \mathbb{R}^{m \times n}$ with a matrix V that has lower precision. For this paper, V has the following structure:

- At its core, V has essentially *integer entries*. We denote this integral matrix by $Z \in \mathbb{Z}^{m \times n}$. The integers are unbounded (not clipped), and later encoded with a good coding scheme. Thus the memory cost of Z is determined by the entropy of the distribution of its entries.
- Additionally, there are real-valued *scaling factors* per row and per column. Concretely, the jth column of V is scaled by $\alpha_j \in \mathbb{R}_{>0}$ and the ith row of V is scaled by $\beta_i \in \mathbb{R}_{>0}$. These input/output scaling factor vectors $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ and $\beta = (\beta_1, \ldots, \beta_m) \in \mathbb{R}^m$ are stored in high precision; their practical memory cost is very small.

So V is an integer matrix Z whose rows and columns have been rescaled by β and α. Concretely, the scaling factors form a rank-1 matrix $\beta\alpha^T \in \mathbb{R}^{m \times n}$, and we have:

$$V = \beta\alpha^T \odot Z \qquad\qquad V_{i,j} = \alpha_j \beta_i \cdot Z_{i,j}$$

In the algorithms, we consider $V_{i,j}$ as implicitly defined as soon as $Z_{i,j}, \alpha_j, \beta_i$ are set.

Crucially, the scaling factors can be used for distributing precision across rows and columns. A small scaling factor corresponds to high precision. For example, if α_1 is very small, that means the scaled integer grid of column 1 is very fine-grained, so column 1 is stored in relatively high precision.

2.2 GPTQ: ADAPTIVE ROUNDING

Algorithm 1 GPTQ

Input: $W \in \mathbb{R}^{m \times n}$, $A = \mathbb{E}[xx^T] \in \mathbb{R}^{n \times n}$, $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}^m$
$L \leftarrow \mathrm{RevCholesky}(A)$
$\tilde{L} \leftarrow L^{-1} = \mathrm{Cholesky}(A^{-1})$
Normalize columns of $\tilde{L}$ by the diagonal entries; i.e., replace $\tilde{L}_{:,j} \leftarrow \tilde{L}_{:,j} / \tilde{L}_{j,j}$ for $j = 1, \ldots, n$.
for $j = 1$ **to** n **do**
  $Z_{:,j} \leftarrow \mathrm{round}(W_{:,j} / (\alpha_j \beta))$
  $\Delta \leftarrow V_{:,j} - W_{:,j}$
  $W \leftarrow W + \Delta \cdot (\tilde{L}_{:,j})^T$
end for
Output: Z

We recall the standard GPTQ algorithm by Frantar et al. (2023), described in Algorithm 1. The idea of this algorithm is to exploit the correlation between the n input channels of the linear module $W \in \mathbb{R}^{m \times n}$. It does so by aiming to minimize the following quadratic distortion:

$$D_A = \frac{1}{mn} \mathbb{E} \|Vx - Wx\|_2^2 = \frac{1}{mn} \|(V - W) \cdot \mathbb{E}[xx^T]^{1/2}\|_F^2 = \frac{1}{mn} \|(V - W) L^T\|_F^2 \qquad (1)$$

The expectation here, and in Section 5, is taken over an (empirical) calibration data distribution, $x \in \mathbb{R}^n$ is an input to the linear module W, and $L = \mathrm{RevCholesky}(\mathbb{E}[xx^T])$. The (reverse) Cholesky

decomposition L encodes linear correlation between the input channels, and sits at the heart of the GPTQ algorithm. The algorithm quantizes the input channels sequentially, and modifies W along the way in order to partially correct the quantization error introduced by quantizing earlier input channels, which is only possible due to the correlation between those channels. See Algorithm 1.

2.3 Two-Sided Extension and Kronecker-Factored Hessians

Later (Kim et al., 2025; Tseng et al., 2026; Birnick & Saab, 2026) it was realized that GPTQ can be naturally extended to also take the output geometry into account. Indeed, the Hessian of the minimization problem $\mathbb{E}\|Vx - Wx\|_2^2/2$ with respect to $v = \mathrm{vec}(V)$ is $\mathbb{E}[xx^T] \otimes I_m$. While the $\mathbb{E}[xx^T]$ factor represents the input correlations that GPTQ exploits, the I_m factor means that output channels are quantized independently. The two-sided version of the algorithm instead considers an arbitrary Kronecker-factored Hessian $A \otimes B$, which corresponds to the following two-sided distortion:

$$\begin{aligned} D_{A,B} &= \frac{1}{mn}\|(B^{1/2})^T(V-W)A^{1/2}\|_F^2 &&= \frac{1}{mn}\|(\mathrm{vec}(V)^T - \mathrm{vec}(W)^T)(A^{1/2} \otimes B^{1/2})\|_F^2 \\ &= \frac{1}{mn}\|L^{(B)}(V-W)(L^{(A)})^T\|_F^2 &&= \frac{1}{mn}\|(\mathrm{vec}(V)^T - \mathrm{vec}(W)^T)(L^{(A)} \otimes L^{(B)})^T\|_F^2 \end{aligned}$$

Here $L^{(A)} = \mathrm{RevCholesky}(A)$ and $L^{(B)} = \mathrm{RevCholesky}(B)$, and since matrix products distribute over Kronecker factors, $L^{(A)} \otimes L^{(B)} = \mathrm{RevCholesky}(A \otimes B)$. The identities from the right column, when compared to Equation (1), already reveal how the natural extension of GPTQ to a two-sided geometry works: We can just call standard GPTQ (Algorithm 1) with $\mathrm{vec}(W)^T \in \mathbb{R}^{1\times mn}$ as the weight matrix and $A \otimes B \in \mathbb{R}^{mn\times mn}$ as the input correlation matrix. This is described in Algorithm 2 in "unvectorized" notation. Note that if $B = I_m$, this is equivalent to Algorithm 1. For fast implementations of Algorithm 2, see Birnick & Saab (2026); Chen et al. (2026a).

Algorithm 2 Two-sided extension of GPTQ

Input: $W \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}^m$
$L^{(A)}, L^{(B)} \leftarrow \mathrm{RevCholesky}(A), \mathrm{RevCholesky}(B)$
$\tilde{L}^{(A)}, \tilde{L}^{(B)} \leftarrow (L^{(A)})^{-1}, (L^{(B)})^{-1}$
Normalize columns of $\tilde{L}^{(A)}, \tilde{L}^{(B)}$ by the diagonal entries.
for $j = 1$ **to** n **do**
  for $i = 1$ **to** m **do**
    $Z_{i,j} \leftarrow \mathrm{round}(W_{i,j}/(\alpha_j\beta_i))$
    $\Delta \leftarrow V_{i,j} - W_{i,j}$
    $W \leftarrow W + \tilde{L}^{(B)}_{:,i} \cdot \Delta \cdot (\tilde{L}^{(A)}_{:,j})^T$
  end for
end for
Output: Z

2.4 Information-Theoretic View and Waterfilling Formula

Note that GPTQ (Algorithm 1) and its two-sided extension (Algorithm 2) are at their core just about error propagation using the (reverse) Cholesky factors L, and are essentially independent of the alphabet that is used to store the entries. In particular, they do not postulate specific scaling factors α, β. Instead, those scaling factors are chosen up front and handed to the algorithm. For example, Huffman-GPTQ (Chen et al., 2026b) implicitly chooses $\alpha \propto (1, \ldots, 1)$ and $\beta \propto (1, \ldots, 1)$, i.e., a single scalar scaling factor for the whole matrix.

As noted in Section 2.1, the scaling factors can be used to distribute precision/rate across rows and columns non-uniformly. Ordentlich & Polyanskiy (2026) realized this and derived information-theoretically optimal scaling factors in the case of standard (only input-sided) GPTQ, i.e., Algorithm 1. Recall that in this setting we have

$$A = \mathbb{E}[xx^T] \qquad L = \mathrm{RevCholesky}(A) \qquad \tilde{L} = L^{-1} = \mathrm{Cholesky}(A^{-1})$$

where the expectation is over calibration data. Ordentlich & Polyanskiy (2026); Lifar et al. (2026) then postulate the input scaling factor choice $\alpha_j \propto 1/L_{j,j} = \tilde{L}_{j,j}$. Since they only consider the standard input-sided algorithm, they implicitly choose uniform output scaling factors $\beta \propto (1, \ldots, 1)$.

They prove that, if $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is modeled as iid Gaussian, then Algorithm 1 with this choice of scaling factors is an (almost) information-theoretically optimal quantization scheme for the distortion in Equation (1). Concretely, they consider arbitrary encoder-decoder pairs: The encoder $f\colon \mathbb{R}^{m\times n} \times \mathbb{R}^{n\times n} \to \{0,1\}^*$ takes (W, A) and produces a self-delimiting (prefix-free) binary string, and the decoder $g\colon \{0,1\}^* \to \mathbb{R}^{m\times n}$ takes the string and produces V, so that we have $V = g(f(W, A))$. For example, one such encoder-decoder pair is given by Algorithm 1 together with an entropy coding scheme to encode/decode Z. (Here we gloss over the encoding of α, β; see Appendix C.1.) For such a pair, the *distortion* D_A is measured as the expected input-sided distortion from Equation (1) and the *rate* R as the expected string length per entry.

$$D_A = \frac{1}{mn}\mathbb{E}_W\left[\|(V-W)\cdot A^{1/2}\|_F^2\right] \qquad R = \frac{1}{mn}\mathbb{E}_W\left[\text{length}(f(W,A))\right] \tag{2}$$

Note that at this point A is seen as a deterministic Gram matrix, and the expectation runs over the Gaussian distribution of W. In this setting, they prove the following theorems (reformulated).

Theorem 2.1 (IT lower bound)**.** *Suppose that $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is iid Gaussian and $A \in \mathbb{R}^{n\times n}$ is positive definite. Then any encoder-decoder pair (f, g) satisfies:*

$$D_A \geq \sigma_W^2 |A|^{1/n} 2^{-2R}$$

Theorem 2.2. *Suppose that $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is iid Gaussian and $A \in \mathbb{R}^{n\times n}$ is positive definite. Additionally fix an entropy coding scheme that encodes Z with expected length $H(Z) + o(mn)$. Then Algorithm 1 with scale choice $\alpha_j = \gamma \cdot |A|^{1/2n}/L_{j,j}$ and $\beta_i = 1$ achieves*

$$D_A = \frac{\gamma^2}{12}|A|^{1/n}\cdot(1+o(1)) \qquad R = \frac{1}{2}\log(2\pi e\sigma_W^2) - \log\gamma + o(1)$$

as $\gamma \to 0$ and $mn \to \infty$. This yields the rate-distortion tradeoff

$$D_A = \frac{2\pi e}{12}\cdot\sigma_W^2|A|^{1/n}2^{-2R}\cdot(1+o(1))$$

which, in the limit, is within $\frac{1}{2}\log(2\pi e/12) \approx 0.255$ bits of the limiting rate from Theorem 2.1.

The scaling factor choice $\alpha_j \propto 1/L_{j,j}$ is known as the high-rate waterfilling rule. Algorithm 1 with these scaling factors, and potentially with other GPTQ improvements on top, is called the "WaterSIC" algorithm, see Lifar et al. (2026).

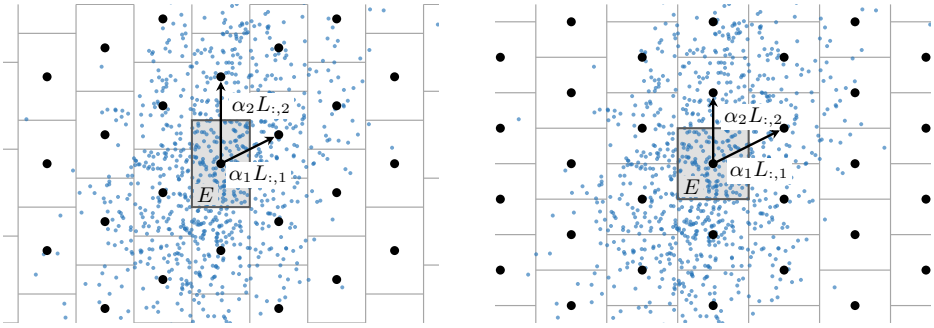

Figure 1: Babai rounding for $L = \left(\begin{smallmatrix} 2 & 0 \\ 1 & 3 \end{smallmatrix}\right)$. Lattice points $L\,\text{diag}(\alpha)\mathbb{Z}^2$ are represented by black dots. Target vectors Lw with $w \sim \mathcal{N}(0, I_2)$ are represented by blue dots. Each target is mapped to the lattice point at the center of its cell. The fundamental region E that contains the quantization errors is highlighted. On the left, the scaling factors are $\alpha = (1, 1)$, giving rectangular cells. On the right, they are chosen optimally as $\alpha = (3/\sqrt{6},\ 2/\sqrt{6})$, giving square cells while preserving the volume.

We will provide some intuition for why $\alpha_j \propto 1/L_{j,j}$ is the correct choice of scaling factors, as this will become important in Section 5. For this, we take a lattice perspective (Birnick, 2026; Chen et al., 2026b): Algorithm 1 is equivalent to running Babai's nearest plane algorithm (Babai, 1986) on the lattice $L\text{diag}(\alpha)\mathbb{Z}^n$ with target vectors $LW_{i,:}^T$. It tiles $\mathbb{R}^n$ with copies of a rectangular fundamental region, where the rectangle has side lengths $(\alpha_1 L_{1,1}, \ldots, \alpha_n L_{n,n})$. Each target vector is quantized

to the lattice vector at the center of the rectangle that it lies inside. Denote the rectangle around the zero vector by $E = \prod_{j=1}^{n}[-\alpha_j L_{j,j}/2, \alpha_j L_{j,j}/2]$. The quantization error vectors $L(V_{i,:} - W_{i,:})^T$ lie in this region. In the high-rate regime (where the scaling factors approach zero), the error vectors are approximately uniformly distributed within E. Thus the distortion is:

$$D_A = \frac{1}{n}\mathbb{E}_{W,i}\left[\|(V_{i,:} - W_{i,:})L^T\|_F^2\right] \approx \frac{1}{n}\mathbb{E}_{e\sim\mathrm{Unif}(E)}\left[\|e\|_2^2\right] = \frac{1}{12}\frac{1}{n}\sum_{j=1}^{n}(\alpha_j L_{j,j})^2 \tag{3}$$

Moreover, fixing the rate R corresponds to fixing the *volume* of the rectangle E. So minimizing distortion at a given rate corresponds to minimizing Equation (3) while leaving $\mathrm{Vol}(E) = \prod_{j=1}^{n}\alpha_j L_{j,j}$ fixed. This is achieved by making the side lengths of E all equal, so that it becomes a cube. This yields $\alpha_j \propto 1/L_{j,j}$. See Figure 1 for a visualization.

3 Two-Sided Rate Allocation

We aim to unify the two-sided Algorithm 2 with the information-theoretic view on Algorithm 1. The first step is to choose the scaling factors α and β for Algorithm 2 appropriately. Perhaps unsurprisingly, we show that the optimal choice is $\alpha_j \propto 1/L^{(A)}_{j,j}$ and $\beta_i \propto 1/L^{(B)}_{i,i}$. This follows directly from viewing Algorithm 2 as being equivalent to Algorithm 1 on a transformed input. We establish the corresponding information-theoretic and achievability theorems for the two-sided Algorithm 2.

Concretely, let $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ iid Gaussian, let the encoder $f\colon \mathbb{R}^{m\times n}\times\mathbb{R}^{n\times n}\times\mathbb{R}^{m\times m} \to \{0,1\}^*$ take (W, A, B), let the decoder $g\colon \{0,1\}^* \to \mathbb{R}^{m\times n}$ be as before, and define rate and distortion in the two-sided setting as follows:

$$D_{A,B} = \frac{1}{mn}\mathbb{E}_W\left[\|(B^{1/2})^T(V - W)\cdot A^{1/2}\|_F^2\right] \qquad R = \frac{1}{mn}\mathbb{E}_W\left[\mathrm{length}(f(W, A, B))\right] \tag{4}$$

Then we have the following results as the two-sided versions of Theorem 2.1 and Theorem 2.2.

Corollary 3.1 (IT lower bound)**.** *Suppose that $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is iid Gaussian and A, B are positive definite. Then any encoder-decoder pair (f, g) satisfies:*

$$D_{A,B} \geq \sigma_W^2 |A|^{1/n}|B|^{1/m}2^{-2R}$$

Proof. Define $\hat{n} := mn$ and $\hat{m} := 1$, as well as $\hat{A} := A\otimes B \in \mathbb{R}^{\hat{n}\times\hat{n}}$ and $\hat{W} := \mathrm{vec}(W)^T \in \mathbb{R}^{\hat{m}\times\hat{n}}$. Then note that iid $\hat{W}_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ and invoke Theorem 2.1 with $\hat{W}$ and $\hat{A}$. Lastly, note that the rate and distortion definitions (2) for $\hat{A}, \hat{W}$ transform exactly to the definitions (4) for A, B, W, e.g., $D_{A,B} = D_{\hat{A}}$, and that $|\hat{A}|^{1/\hat{n}} = |A\otimes B|^{1/(mn)} = (|A|^m|B|^n)^{1/(mn)} = |A|^{1/n}|B|^{1/m}$. So the conclusion of our invocation of Theorem 2.1 is exactly the conclusion of this corollary. □

Corollary 3.2. *Suppose that $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is iid Gaussian and A, B are positive definite. Additionally fix an entropy coding scheme that encodes Z with expected length $H(Z) + o(mn)$. Then Algorithm 2 with scale choice $\alpha_j = \sqrt{\gamma}\cdot|A|^{1/2n}/L^{(A)}_{j,j}$ and $\beta_i = \sqrt{\gamma}\cdot|B|^{1/2m}/L^{(B)}_{i,i}$ achieves*

$$D_{A,B} = \frac{\gamma^2}{12}|A|^{1/n}|B|^{1/m}\cdot(1 + o(1)) \qquad R = \frac{1}{2}\log(2\pi e\sigma_W^2) - \log\gamma + o(1)$$

as $\gamma \to 0$ and $mn \to \infty$. This yields the rate-distortion tradeoff

$$D_{A,B} = \frac{2\pi e}{12}\cdot\sigma_W^2|A|^{1/n}|B|^{1/m}2^{-2R}\cdot(1 + o(1))$$

which, in the limit, is within $\frac{1}{2}\log(2\pi e/12) \approx 0.255$ bits of the limiting rate from Corollary 3.1.

Proof. We again define $\hat{n} := mn$ and $\hat{m} := 1$, then $\hat{A} := A\otimes B \in \mathbb{R}^{\hat{n}\times\hat{n}}$ and $\hat{W} := \mathrm{vec}(W)^T \in \mathbb{R}^{\hat{m}\times\hat{n}}$, and also $\hat{\alpha} = \alpha\otimes\beta \in \mathbb{R}^{\hat{n}}$ and $\hat{\beta} = 1 \in \mathbb{R}^{\hat{m}}$. Define $\hat{L}, L^{(A)}, L^{(B)}$ as the reverse Cholesky decompositions of $\hat{A}, A, B$, and note that $\hat{L} = L^{(A)}\otimes L^{(B)}$. Here, again, we have iid $\hat{W}_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ and we invoke Theorem 2.2 with $\hat{W}$ and $\hat{A}$.

Now note that calling Algorithm 1 with $\hat{W}, \hat{A}, \hat{\alpha}, \hat{\beta}$, upon unvectorizing the resulting $\hat{V}$ (or $\hat{Z}$) back into an m by n matrix, is equivalent to calling Algorithm 2 with W, A, B, α, β. And if, for any pair (i, j), we define $\hat{j} := (j-1)m + i$, then we have

$$\hat{\alpha}_{\hat{j}} = (\alpha \otimes \beta)_{\hat{j}} = \alpha_j \beta_i = \sqrt{\gamma} \cdot |A|^{1/2n} / L^{(A)}_{j,j} \cdot \sqrt{\gamma} \cdot |B|^{1/2m} / L^{(B)}_{i,i} = \gamma \cdot |\hat{A}|^{1/2\hat{n}} / \hat{L}_{\hat{j},\hat{j}}$$

which shows that the scaling factors of this corollary correspond to those prescribed by our invocation of Theorem 2.2. Again, the rate and distortion definitions (2) and (4) match, $D_{A,B} = D_{\hat{A}}$, so the conclusion of our invocation of Theorem 2.2 is exactly the conclusion of this corollary. □

Remark. The only subtlety is the entropy coding scheme for Z. For the algorithm from Theorem 2.2, Lifar et al. (2026) suggest Huffman coding for every column, which is possible since all entries in one column j have the same scaling factor α_j, due to the row scaling factors $\beta = (1, \dots, 1)$ being trivial. For the algorithm in this corollary, this kind of Huffman coding is no longer efficient enough since both α and β are nontrivial, and thus every entry (i, j) has a unique scaling factor $\alpha_j \beta_i$. However, other entropy coding schemes can deal with this, such as variants of arithmetic coding.

4 Kronecker-Hessian Mismatch Factor

So far, we have analyzed the two-sided algorithm with the distortion measure $D_{A,B}$, which implicitly assumes a Kronecker-factored Hessian structure $A \otimes B$. However, in reality, the Kronecker product $A \otimes B$ is only an *approximation* to some "full Hessian" $H \in \mathbb{R}^{mn \times mn}$ that truly captures the geometry of the problem, but is too big to directly deal with algorithmically. In practice, H is usually the Fisher information matrix of W, see Section 5. The true distortion must be measured with respect to H, which leads to the following distortion measure:

$$D_H = \frac{1}{mn} \mathbb{E}_W \left[\| (\mathrm{vec}(V)^T - \mathrm{vec}(W)^T) \cdot H^{1/2} \|_F^2 \right] \tag{5}$$

We now show that when approximating H by $A \otimes B$ and running Algorithm 2 with A, B, the full distortion D_H asymptotically equals the two-sided distortion $D_{A,B}$ from Corollary 3.2 times $\mathrm{tr}(H(A \otimes B)^{-1})/mn$. Note that it is not generally true that $D_H = D_{A,B} \cdot \mathrm{tr}(H(A \otimes B)^{-1})/mn$. In general there is a correction term, which for Algorithm 2 in the setting of our theorem becomes negligible as the quantization errors in E become asymptotically uniformly distributed.

Theorem 4.1. *Suppose that $W_{i,j} \sim \mathcal{N}(0, \sigma_W^2)$ is iid Gaussian and H, A, B are positive definite. Additionally fix an entropy coding scheme that encodes Z with expected length $H(Z) + o(mn)$. Then Algorithm 2 with scale choice $\alpha_j = \sqrt{\gamma} \cdot |A|^{1/2n} / L^{(A)}_{j,j}$ and $\beta_i = \sqrt{\gamma} \cdot |B|^{1/2m} / L^{(B)}_{i,i}$ achieves*

$$D_H = \frac{\gamma^2}{12} |A|^{1/n} |B|^{1/m} \frac{\mathrm{tr}(H(A \otimes B)^{-1})}{mn} \cdot (1 + o(1)) \qquad R = \frac{1}{2} \log(2\pi e \sigma_W^2) - \log \gamma + o(1)$$

as $\gamma \to 0$ and $mn \to \infty$. This yields the rate-distortion tradeoff:

$$D_H = \frac{2\pi e}{12} \cdot \sigma_W^2 |A|^{1/n} |B|^{1/m} \frac{\mathrm{tr}(H(A \otimes B)^{-1})}{mn} 2^{-2R} \cdot (1 + o(1))$$

Proof. The setting is as in Corollary 3.2, which already provides the rate $R = \frac{1}{2} \log(2\pi e \sigma_W^2) - \log \gamma + o(1)$. We only have to compute the distortion D_H with respect to the full Hessian. Recall the intuition from Section 2.4. For the two-sided algorithm, the "error region" in the lattice space is

$$E := \prod_{j=1}^{n} \prod_{i=1}^{m} [-\alpha_j \beta_i L^{(A)}_{j,j} L^{(B)}_{i,i} / 2, \alpha_j \beta_i L^{(A)}_{j,j} L^{(B)}_{i,i} / 2]$$

The scaling factors are chosen exactly so that this becomes a cube. Concretely, we have:

$$E = |A|^{1/2n} |B|^{1/2m} \cdot \prod_{j=1}^{n} \prod_{i=1}^{m} [-\gamma/2, \gamma/2]$$

The transformed quantization error $\mathrm{vec}(L^{(B)}(V - W)(L^{(A)})^T)$ lies in E. For simplicity, we work only in the high-rate limit $\gamma \to 0$ and assume that this transformed quantization error vector is uniformly distributed in E. As $\gamma \to 0$ this holds asymptotically because W is normally distributed; intuitively, in Figure 1, the rectangles/cubes become so small that the blue dots become approximately

uniformly distributed within them. For a concrete proof of this, see Lifar et al. (2026, Appendix B.2). Write $L := L^{(A)} \otimes L^{(B)}$, $d := \mathrm{vec}(V - W)$, and note that $e := Ld$ is the transformed quantization error $\mathrm{vec}(L^{(B)}(V - W)(L^{(A)})^T) \in E$. So $d = L^{-1}e$, where e is asymptotically uniformly distributed, giving:

$$\begin{aligned} D_H &= \frac{1}{mn}\mathbb{E}_W[d^T H d] = \frac{1}{mn}\mathbb{E}_W[\mathrm{tr}(Hdd^T)] = \frac{1}{mn}\mathbb{E}_W[\mathrm{tr}(HL^{-1}ee^TL^{-T})] \\ &\overset{\gamma\to 0}{\asymp} \frac{1}{mn}\,\mathrm{tr}(HL^{-1}\mathbb{E}_{e\sim\mathrm{Unif}(E)}[ee^T]L^{-T}) = \frac{1}{mn}\frac{|A|^{1/n}|B|^{1/m}\gamma^2}{12}\,\mathrm{tr}(HL^{-1}IL^{-T}) \\ &= \frac{\gamma^2}{12}|A|^{1/n}|B|^{1/m}\frac{\mathrm{tr}(H(A\otimes B)^{-1})}{mn} \end{aligned}$$

□

Theorem 4.1 now gives us a clear criterion for choosing the Hessian approximation $A \otimes B$ in order to minimize distortion: We have to minimize the term in the distortion that depends on A, B, which is $|A|^{1/n}|B|^{1/m}\,\mathrm{tr}(H(A\otimes B)^{-1})/mn$. It is cleanest to additionally normalize this by $|H|^{1/mn}$. We call the resulting metric the *Kronecker-Hessian mismatch factor* and abbreviate it as Φ:

$$\Phi(H, A\otimes B) := \frac{\mathrm{tr}(H(A\otimes B)^{-1})/mn}{|H|^{1/mn}\cdot|A|^{-1/n}|B|^{-1/m}} = \frac{\mathrm{tr}(H(A\otimes B)^{-1})/mn}{|H|^{1/mn}\cdot|A\otimes B|^{-1/mn}}$$

The mismatch factor has the following additional interpretations. First, define the *generalized eigenvalues* $\lambda_1, \ldots, \lambda_{mn}$ of $(H, A\otimes B)$ as the eigenvalues of $((A\otimes B)^{1/2})^{-1}H((A\otimes B)^{1/2})^{-T}$. Then Φ is the ratio of the arithmetic mean to the geometric mean of those generalized eigenvalues:

$$\Phi(H, A\otimes B) = \frac{\frac{1}{mn}\sum_{k=1}^{mn}\lambda_k}{(\prod_{k=1}^{mn}\lambda_k)^{1/mn}} = \frac{\mathrm{AM}(\lambda_1,\ldots,\lambda_{mn})}{\mathrm{GM}(\lambda_1,\ldots,\lambda_{mn})}$$

The AM-GM inequality gives $\Phi(H, A\otimes B) \geq 1$ with equality if and only if $H \propto A\otimes B$. Φ is also related to the *coding gain* in transform coding, see Gersho & Gray (2012, Chapter 8).

Second, up to scaling $A\otimes B$ by some $s > 0$, it is essentially equal to the exponential of the KL divergence between Gaussians with covariances H and $A\otimes B$, respectively:

$$\log\Phi(H, A\otimes B) = \frac{2}{mn}\min_{s>0} D_{\mathrm{KL}}(\mathcal{N}(0,H)\,\|\,\mathcal{N}(0, s\cdot A\otimes B)) \tag{6}$$

5 The FlipFlop Hessian Approximation

In order to minimize the distortion in Theorem 4.1, we have to find a Hessian approximation $A\otimes B$ of H that minimizes $\Phi(H, A\otimes B)$. In general, this minimization problem has no closed-form solution. But it turns out that there is an efficient algorithm that converges to the solution. We work in the setting where H is an expectation over outer products of Kronecker-factored vectors:

$$H = \mathbb{E}_{x,g}[(x\otimes g)(x\otimes g)^T] = \mathbb{E}_{x,g}[xx^T\otimes gg^T]$$

Here $x \in \mathbb{R}^n$ and $g \in \mathbb{R}^m$, and we will drop the subscript x, g from the expectation for the remainder of the section. There is a very natural setting where this is the case, namely when H is the Fisher information matrix of W, or equivalently the Hessian of the cross-entropy loss with respect to the network's own predictive distribution. In that case, x is a calibration input of W with output $y = Wx$, a network output is sampled from the model's predictive distribution, and g is the gradient of the negative log-likelihood of this network output with respect to y (see, e.g., Birnick & Saab, 2026).

We now wish to find A, B that minimize $\Phi(H, A\otimes B)$. Write $G := gx^T \in \mathbb{R}^{m\times n}$ and note that $\mathrm{vec}(G) = x\otimes g$. From trace and Kronecker identities we get:

$$\mathrm{tr}(H(A\otimes B)^{-1}) = \mathrm{tr}(\mathbb{E}[\mathrm{vec}(G)\,\mathrm{vec}(G)^T](A^{-1}\otimes B^{-1})) = \mathbb{E}\,\mathrm{tr}(B^{-1}GA^{-1}G^T)$$

Thus we can rewrite the objective Φ, and also simplify it by removing constants:

$$\Phi(H, A\otimes B) \quad\propto\quad |A|^{1/n}|B|^{1/m}\cdot\mathbb{E}\,\mathrm{tr}(B^{-1}GA^{-1}G^T)$$

Now we fix A, and ask for the best choice of B given A. The objective for B is proportional to

$$|B|^{1/m}\cdot\mathrm{tr}(B^{-1}S) \quad\text{with}\quad S := \mathbb{E}[GA^{-1}G^T].$$

This is minimized when B is proportional to S. Indeed, the AM-GM inequality applied to the eigenvalues of $B^{-1}S$ says $\mathrm{tr}(B^{-1}S)/m \geq (|S|/|B|)^{1/m}$ and equality holds when the eigenvalues are all the same, leading to $B \propto S$. Similarly, when B is fixed, then A minimizes Φ precisely when $A \propto \mathbb{E}[G^T B^{-1} G]$. This naturally leads to the following alternating (fixed-point) iteration:

$$A_{k+1} \propto \mathbb{E}[G^T B_k^{-1} G] = \mathbb{E}[g^T B_k^{-1} g \cdot xx^T]$$
$$B_{k+1} \propto \mathbb{E}[G A_{k+1}^{-1} G^T] = \mathbb{E}[x^T A_{k+1}^{-1} x \cdot gg^T]$$

In maximum likelihood estimation of matrix normal distributions, this is known as the "flip-flop algorithm" (Dutilleul, 1999). If H is positive definite, it converges to a unique (up to scaling) fixed point (Hoff et al., 2023, Section 2). The connection to Gaussian maximum likelihood estimation is given by Equation (6). Hence we call the resulting approximation $A \otimes B$ of H the *FlipFlop* Hessian.

Interestingly, the algorithm resembles a power iteration that was previously explored for two-sided GPTQ. Namely, if A_{k+1}^{-1}, B_k^{-1} above are replaced by A_{k+1}, B_k, so that the weighting is in some sense reciprocal, then the algorithm becomes a power iteration that minimizes $\|H - A \otimes B\|_F$, see Tseng et al. (2026); Birnick & Saab (2026). We call the resulting approximation the *Frobenius* Hessian. It can be interpreted as the best "additive" Kronecker approximation to H, while the FlipFlop Hessian is, in a certain sense, the best "multiplicative" approximation. The FlipFlop Hessian outperforms the Frobenius Hessian in practice, as suggested by the theory, see Appendix B.

6 Experiment Findings

We validate our theory empirically by quantizing transformer language models with the Hessian approximations from Table 1. For the concrete setup and result tables, see Appendix B. We find that generally the two-sided ("WaterKron") algorithm outperform the purely input-sided ("WaterSIC") algorithm, *if the Hessian approximation is chosen carefully* like argued in the paper. The Frobenius variant, as used for example by YAQA (Tseng et al., 2026), generally gets *worse* with more iterations (measured up to 3 iterations). For the larger 8B models it even gets clearly worse than the purely input-sided Hessian. This supports the argument against minimizing $\|H - A \otimes B\|_F$. Our new FlipFlop variant generally *improves* with the second iteration, and is consistently the best of all the variants, supporting the argument that $\Phi(H, A \otimes B)$ is a more appropriate metric to minimize.

Table 1: Different Hessian approximations $A \otimes B$ of $H = \mathbb{E}[xx^T \otimes gg^T]$. Frobenius and FlipFlop approximations are computed using the alternating fixed-point iteration, starting from the identity matrix.

Name	Input factor A	Output factor B	Notes
Input	$\mathbb{E}[xx^T]$	I_m	Recovers WaterSIC.
Marginal	$\mathbb{E}[xx^T]$	$\mathbb{E}[gg^T]$	Treats x, g as independent.
Frobenius	$\mathbb{E}[(g^T B g)\, xx^T]$	$\mathbb{E}[(x^T A x)\, gg^T]$	Targets $\|H - A \otimes B\|_F$.
FlipFlop	$\mathbb{E}[(g^T B^{-1} g)\, xx^T]$	$\mathbb{E}[(x^T A^{-1} x)\, gg^T]$	Targets $\Phi(H, A \otimes B)$.

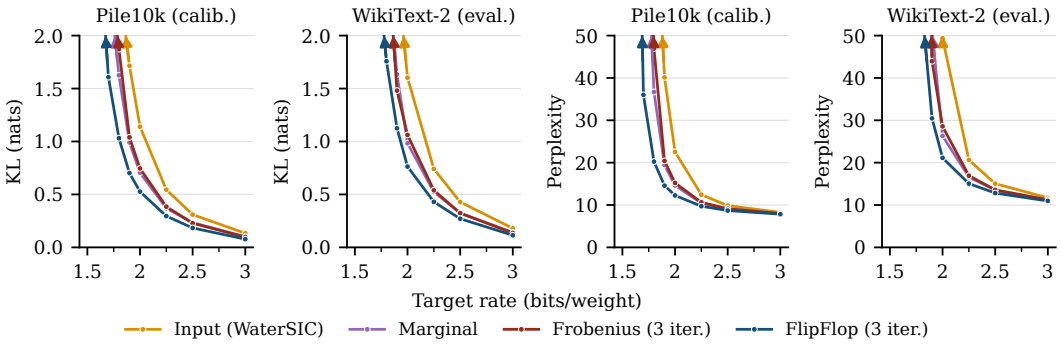

Figure 2: Llama-3.2-1B KL/PPL across rates for various Hessian approximations. (See Figure 3.)

AI USE STATEMENT

In this work, generative AI has been used in research discussion to help develop theoretical models or conceptual frameworks like the Hessian mismatch theory from Section 4, in particular to propose or refine hypotheses and formulate mathematical claims. It has also been used to help implement those methods in software code for experimentation.

Generative AI has *not* been used to write the paper itself, except to format tables, help to create figures, and for other polishing work like finding typos. It has also not been used to provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, or to interpret results. Similarly, it was not used for tasks that are not applicable in this paper, such as to generate synthetic data sets, assist with translation, clean and reformat datasets, or to support qualitative and thematic data analysis.

We have reviewed all AI-assisted work. In particular, we verified all generated mathematical claims and reviewed all generated software code. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The source code to reproduce the experiments will be made public with the publication of the paper.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6 (1):1–13, 1986.

Johann Birnick. The lattice geometry of neural network quantization: A short equivalence proof of GPTQ and Babai's algorithm. In *International Conference on Learning Representations*, volume 2026, pp. 101779–101787, 2026.

Johann Birnick and Rayan Saab. FastKron: Efficient quantization with Kronecker-factored Hessians, 2026. URL `https://arxiv.org/abs/2608.06291`.

Joao V Cavalcanti and Ashia C Wilson. SoftWater: Class-aware rate allocation for softmax quantization. *arXiv preprint arXiv:2608.12026*, 2026.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. QuIP: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

Jiale Chen, Torsten Hoefler, and Dan Alistarh. GPTQ-2D: Cubic-time two-sided adaptive rounding. *arXiv preprint arXiv:2607.27042*, 2026a.

Jiale Chen, Yalda Shabanzadeh, Elvir Crnčević, Torsten Hoefler, and Dan Alistarh. The geometry of LLM quantization: GPTQ as Babai's nearest plane algorithm. In *International Conference on Learning Representations*, volume 2026, pp. 122653–122695, 2026b.

Pierre Dutilleul. The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 2012.

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.

Peter Hoff, Andrew McCormack, and Anru R Zhang. Core shrinkage covariance estimation for matrix-variate data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85 (5):1659–1679, 2023.

Junhan Kim, Ho-young Kim, Eulrang Cho, Chungman Lee, Joonyoung Kim, and Yongkweon Jeon. BoA: Attention-aware post-training quantization without backpropagation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Uvj6XcSJ5d.

Egor Lifar, Semyon Savkin, Or Ordentlich, and Yury Polyanskiy. WaterSIC: Information-theoretically (near) optimal linear layer quantization. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=fCPgAHIciE.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Or Ordentlich and Yury Polyanskiy. High-rate quantized matrix multiplication II, 2026. URL https://arxiv.org/abs/2605.13768.

Semyon Savkin, Eitan Porat, Or Ordentlich, and Yury Polyanskiy. NestQuant: Nested lattice quantization for matrix products and LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4OWGON33HE.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. *Proceedings of Machine Learning Research*, 235:48630, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. *Advances in Neural Information Processing Systems*, 37:59597–59620, 2024b.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=PKFilPWjMI.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Haoyu Zhang, Shihao Zhang, Ian Colbert, and Rayan Saab. Provable post-training quantization: Theoretical analysis of OPTQ and Qronos. *arXiv preprint arXiv:2508.04853*, 2025.

Shihao Zhang, Haoyu Zhang, Ian Colbert, and Rayan Saab. Qronos: Correcting the past by shaping the future... in post-training quantization. In *International Conference on Learning Representations*, volume 2026, pp. 87775–87798, 2026.

A LIMITATIONS

The main limitation for the practical use of WaterKron is carried over from Huffman-GPTQ and WaterSIC, namely the use of entropy coding. Decoders for entropy coding schemes may be hard to implement efficiently as GPU kernels. Practical workarounds include the use of compression schemes that are natively supported on the accelerator, or splitting Z into a dense matrix of a fixed datatype and a sparse matrix of outliers.

Additionally, the theory works (naturally) in the high-rate regime, while practical neural network quantization and the experiments are in the low-rate regime.

B EXPERIMENT RESULTS

Scope. The goal of the experiments is to *isolate the WaterKron and FlipFlop Hessian contribution* by comparing it to the basic WaterSIC algorithm (i.e., the basic GPTQ algorithm with the waterfilling scaling factor choice) in a simple and clean setting without any other algorithmic refinements. As noted already in Section 1, several orthogonal improvements of GPTQ have been developed, which would likely improve the empirical results here. These techniques include drift correction (e.g., Zhang et al., 2026), the use of improved base quantizers (e.g., Savkin et al., 2025), finetuning the scaling factors, and combination with random/learned rotations; also see the list of techniques applied in Lifar et al. (2026). We intentionally do *not* apply any of those techniques, but isolate the role of the Hessian approximation by comparing in a basic setting.

Runs. We quantize and evaluate Llama (Grattafiori et al., 2024) and Qwen (Yang et al., 2025) models with up to 8B parameters at a target rate of 2 bits per weight, see Tables 2 to 7. Additionally, we quantize Llama-3.2-1B at various rates, see Figure 3.

Setup. We only quantize linear modules from the transformer blocks, not the embedding or head matrix. Calibration uses 256 sequences of 2048 tokens from Pile10k, a subset of The Pile (Gao et al., 2020). Inputs x are always taken from the unquantized model, in contrast to using an $\hat{x}$ that went through the partially quantized model, as is usually done with GPTQ. The gradients are computed in an empirical Fisher fashion, meaning that labels are not sampled from the model's predictive distribution but taken as the next token from the calibration dataset. Hessians are damped with 0.1 times the mean diagonal for both factors, including between iterations of the flip-flop algorithm. The Hessian approximations that we compare are listed in Table 1. For Frobenius and FlipFlop approximations, which both use the alternating fixed-point iteration, we additionally compare different numbers of iterations. A is initialized as I_n, and then one iteration corresponds to updating both B and A once. The number of iterations is appended as a suffix to the name, e.g., "FlipFlop-1", "FlipFlop-2", etc. We evaluate $D_{\mathrm{KL}}(p_{\mathrm{unquantized}} \| p_{\mathrm{quantized}})$ (KL) and perplexity (PPL) on both the calibration dataset (Pile10k) and the WikiText-2 test split (Merity et al., 2016), averaged over tokens.

Table 2: Llama-3.2-1B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	7.3	0.00	9.8
Input (WaterSIC)	1.14	22.5	1.60	49.3
Marginal	0.70	14.6	0.98	26.3
Frobenius-1	0.62	13.5	0.91	24.4
Frobenius-2	0.74	15.2	1.09	29.4
Frobenius-3	0.75	15.2	1.06	28.6
FlipFlop-1	0.57	12.7	0.83	22.6
FlipFlop-2	0.53	12.3	**0.76**	**21.0**
FlipFlop-3	**0.53**	**12.3**	0.76	21.1

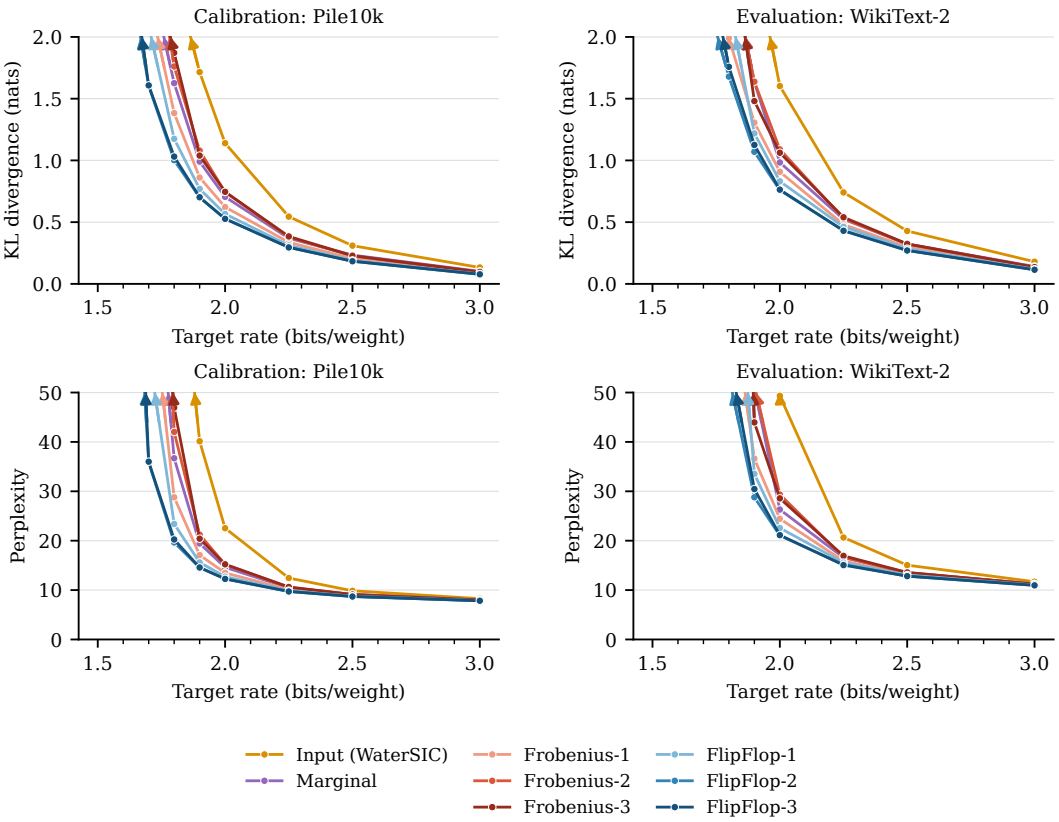

Figure 3: More detailed version of Figure 2. Llama-3.2-1B quantized with all eight Hessian variants across various rates. "FlipFlop" outperforms all other Hessian approximations.

Table 3: Llama-3.2-3B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	6.2	0.00	7.8
Input (WaterSIC)	0.46	9.5	0.72	15.8
Marginal	0.41	9.2	0.71	15.6
Frobenius-1	0.39	9.0	0.68	15.2
Frobenius-2	0.46	9.6	0.82	17.5
Frobenius-3	0.45	9.5	0.80	17.2
FlipFlop-1	0.35	8.6	0.62	14.4
FlipFlop-2	**0.33**	**8.5**	**0.57**	**13.7**
FlipFlop-3	0.33	8.5	0.58	13.8

Table 4: Llama-3.1-8B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	5.3	0.00	6.2
Input (WaterSIC)	0.41	7.9	0.62	11.6
Marginal	0.40	7.9	0.64	11.9
Frobenius-1	0.42	8.0	0.67	12.2
Frobenius-2	0.47	8.4	0.69	12.5
Frobenius-3	0.47	8.4	0.68	12.3
FlipFlop-1	0.35	7.5	0.60	11.4
FlipFlop-2	0.34	7.3	0.55	10.9
FlipFlop-3	**0.34**	**7.3**	**0.54**	**10.8**

Table 5: Qwen3-1.7B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	9.1	0.00	16.7
Input (WaterSIC)	0.78	17.2	1.11	37.2
Marginal	0.74	16.1	1.09	36.1
Frobenius-1	0.64	14.2	0.90	28.8
Frobenius-2	0.76	15.7	1.07	32.3
Frobenius-3	0.77	15.6	1.08	32.5
FlipFlop-1	0.49	12.5	0.76	25.6
FlipFlop-2	0.46	12.3	**0.68**	24.0
FlipFlop-3	**0.46**	**12.3**	0.69	**23.8**

Table 6: Qwen3-4B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	7.8	0.00	13.7
Input (WaterSIC)	0.39	10.1	0.61	19.7
Marginal	0.36	9.9	0.58	19.2
Frobenius-1	0.42	10.2	0.61	18.9
Frobenius-2	0.47	10.6	0.66	19.9
Frobenius-3	0.47	10.8	0.66	19.9
FlipFlop-1	0.34	9.7	0.54	18.8
FlipFlop-2	0.33	9.6	0.53	18.6
FlipFlop-3	**0.33**	**9.6**	**0.52**	**18.3**

Table 7: Qwen3-8B at a target rate of 2 bits per weight.

	Pile10k		WikiText-2	
Hessian	KL ↓	PPL ↓	KL ↓	PPL ↓
Unquantized	0.00	6.5	0.00	9.7
Input (WaterSIC)	0.28	7.9	0.45	13.0
Marginal	0.29	8.0	0.47	13.1
Frobenius-1	0.31	8.0	0.49	13.6
Frobenius-2	0.34	8.2	0.50	13.6
Frobenius-3	0.34	8.2	0.49	13.4
FlipFlop-1	0.26	7.8	0.45	13.0
FlipFlop-2	0.25	**7.7**	**0.42**	**12.7**
FlipFlop-3	**0.25**	7.7	0.42	12.7

C Technical Details

For the theorems in this paper, there are two technicalities that we deliberately left without comment in order to focus on the core statements. In this section, we clarify these technicalities.

C.1 Encoding of the Scaling Factors α, β

Theorem 2.2, Corollary 3.2, and Theorem 4.1 only consider the encoding of $Z \in \mathbb{Z}^{m\times n}$ and ignore the encoding of the scaling factors $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$. This is to keep the theorem statements clean. In practice, the memory cost of the scaling factors is negligible. For the theoretical coding argument, we can give the decoder direct access to α, β, so there is no need to encode them. In fact, the decoder can even be given full access to A, B (which in particular determine α, β), as the lower bounds of Theorem 2.1 and Corollary 3.1 still hold true even with this more powerful decoder.

Concretely, the decoder g would then be a function $g\colon \{0,1\}^* \times \mathbb{R}^{n\times n} \times \mathbb{R}^{m\times m} \to \mathbb{R}^{m\times n}$, taking a string and A, B, and producing V. For Algorithm 2, the string would be the entropy-coded Z, and α, β as defined in Corollary 3.2 can be found from A, B, so the decoder can find $V = \beta\alpha^T \odot Z$. Now, with this more powerful decoder, one could worry that the lower bound of Corollary 3.1 would not hold anymore, as a decoder with access to A, B perhaps allows for a much more efficient scheme; but this is not the case. Indeed, the standard proof of Corollary 3.1 already assumes decoder access to A and B, see Ordentlich & Polyanskiy (2026); Lifar et al. (2026).

C.2 Limit $\gamma \to 0$ and $mn \to \infty$

Theorem 2.2, Corollary 3.2, and Theorem 4.1 all consider the limit $\gamma \to 0$ and $mn \to \infty$. The limit $\gamma \to 0$ makes sure that we are in the high-rate regime where the transformed quantization errors become uniformly distributed. The limit $mn \to \infty$, i.e., growing matrix size, is to ensure the efficiency of the coding scheme, so that it can approach the entropy of the distribution of Z. Note that due to the second limit, these theorems are technically not about a fixed pair A, B, but about all of them at the same time. In this case, one needs to be careful about how the joint limit $\gamma \to 0$ and $mn \to \infty$ is taken. We understand it in an iterated sense: First take $\gamma \to 0$ for fixed A, B, then take $mn \to \infty$. Then for each concrete instance of A, B it is ensured that we are in the high-rate limit where the transformed quantization errors are uniformly distributed.