WaterSIC: Weight-Only Post-Training Quantization of Large Language Models

by

Egor Lifar

S.B. in Computer Science and Engineering and Mathematics, MIT, 2025

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2026

Authored by: Egor Lifar
Department of Electrical Engineering and Computer Science
May 13, 2026

Certified by: Yury Polyanskiy
Professor of Electrical Engineering and Computer Science, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

WaterSIC: Weight-Only Post-Training Quantization of Large Language Models

by

Egor Lifar

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2026 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Large language models incur substantial inference cost from their dense linear layers, where memory bandwidth, not arithmetic, is typically the bottleneck during autoregressive generation. The standard remedy is post-training quantization (PTQ), and most modern PTQ algorithms reduce to a common layerwise problem: given an original weight matrix and a calibration set of input activations, find a low-bit approximation that minimizes the expected mean-squared error of the layer's output. Despite an active literature of such algorithms, how close any of them comes to the information-theoretic limit on this tradeoff has not previously been addressed.

This thesis takes the information-theoretic view. We formulate weight-only quantization of a linear layer as a rate–distortion problem with side information given by the covariance of the input activations, and derive a waterfilling lower bound on the achievable rate–distortion tradeoff that holds uniformly over all input covariance matrices. We show that the popular GPTQ algorithm may have an arbitrarily large rate gap to this limit, depending on the conditioning of the input covariance.

To close this gap, we propose *WaterSIC*, a layerwise quantizer whose key idea is to allocate different quantization rates to different columns (in-features) of the weight matrix, mimicking the classical information-theoretic solution known as reverse waterfilling. We prove that WaterSIC achieves a rate gap of at most ≈ 0.255 bits to the information-theoretic lower bound, uniformly over all input covariance matrices and at all rates: the standard gap between scalar uniform quantization and asymptotically optimal vector quantization. The algorithm extends naturally to attention projections via attention-weighted calibration with adaptive covariance mixing, and to a finetuning variant (WaterSIC-FT) that recovers further accuracy at very low bitrates.

We evaluate WaterSIC on the Llama and Qwen families across model sizes from 1B to 70B parameters and rates from 1 to 4 bits per weight, establishing new state-of-the-art results on WikiText-2 perplexity throughout this range. On a suite of zero-shot accuracy benchmarks, WaterSIC matches or exceeds prior methods on the majority of tasks at every tested bitrate, with the largest gains concentrated in the aggressive low-bit regime. Together, these contributions connect a sharp information-theoretic limit on weight-only PTQ to a practical algorithm that approaches it on deployed LLMs.

Thesis supervisor: Yury Polyanskiy
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Yury Polyanskiy. His mentorship, sharp insights, and the many ideas he shared throughout this project were essential to everything that follows. It is a privilege to have learned from a researcher of his caliber.

I would also like to thank my collaborators on the WaterSIC paper, Semyon Savkin and Prof. Or Ordentlich. The theoretical core of this thesis owes a particular debt to Prof. Ordentlich and Prof. Polyanskiy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The backbone of every large language model is the linear layer: given a column X of input activations, it outputs $Y = WX$ for a weight matrix W. These layers (the attention projections w_q, w_k, w_v, w_o and the feed-forward expansions $w_{\text{up}}, w_{\text{gate}}, w_{\text{down}}$) account for the vast majority of both parameters and inference FLOPs in modern transformers, and are the central technical lever in making LLMs cheaper to serve.

Weight compression matters for two reasons. The first is memory footprint: a 70B-parameter model in BF16 requires roughly 140 GB, exceeding the HBM capacity of a single accelerator. The second, and often more important, is memory *bandwidth*: during autoregressive generation each generated token requires loading the full weight matrix from DRAM, and this transfer typically dominates wall-clock latency rather than arithmetic [4]. Reducing bits per weight therefore cuts both memory consumption and inference latency, with the latter effect most pronounced in the decoding-dominated regime where modern LLMs spend most of their compute.

The goal of *post-training quantization* (PTQ) is to replace a pretrained weight matrix W with a low-precision approximation $\hat{W}$ that can be stored using fewer bits per entry, so that $\hat{Y} = \hat{W}X \approx Y$ for typical inputs X. Activations remain in BF16 or FP16; only the weights are quantized. This *weight-only* regime is the setting of essentially all currently deployed LLM compression schemes, for several reasons: activations exhibit large dynamic ranges and outliers that complicate aggressive activation quantization, the KV-cache is itself an activation tensor whose quantization is an orthogonal problem, and most hardware kernels for low-bit matrix multiplication target the asymmetric weight-low / activation-high regime [5]. It is the setting of this thesis.

While dozens of PTQ algorithms have been proposed in the past few years, most reduce to a common elementary building block, the *layerwise quantizer*: an algorithm that, given a layer's weight matrix W and a calibration distribution over X, produces a quantized $\hat{W}$ minimizing the expected squared error $\mathbb{E}\|(W - \hat{W})X\|^2$ of the layer output [3,6]. Despite this very active line of research, the question of how close any of these algorithms comes to the information-theoretic limit on the achievable rate-distortion tradeoff has not previously been addressed. Establishing that limit, and giving an algorithm with a provably small gap to it,

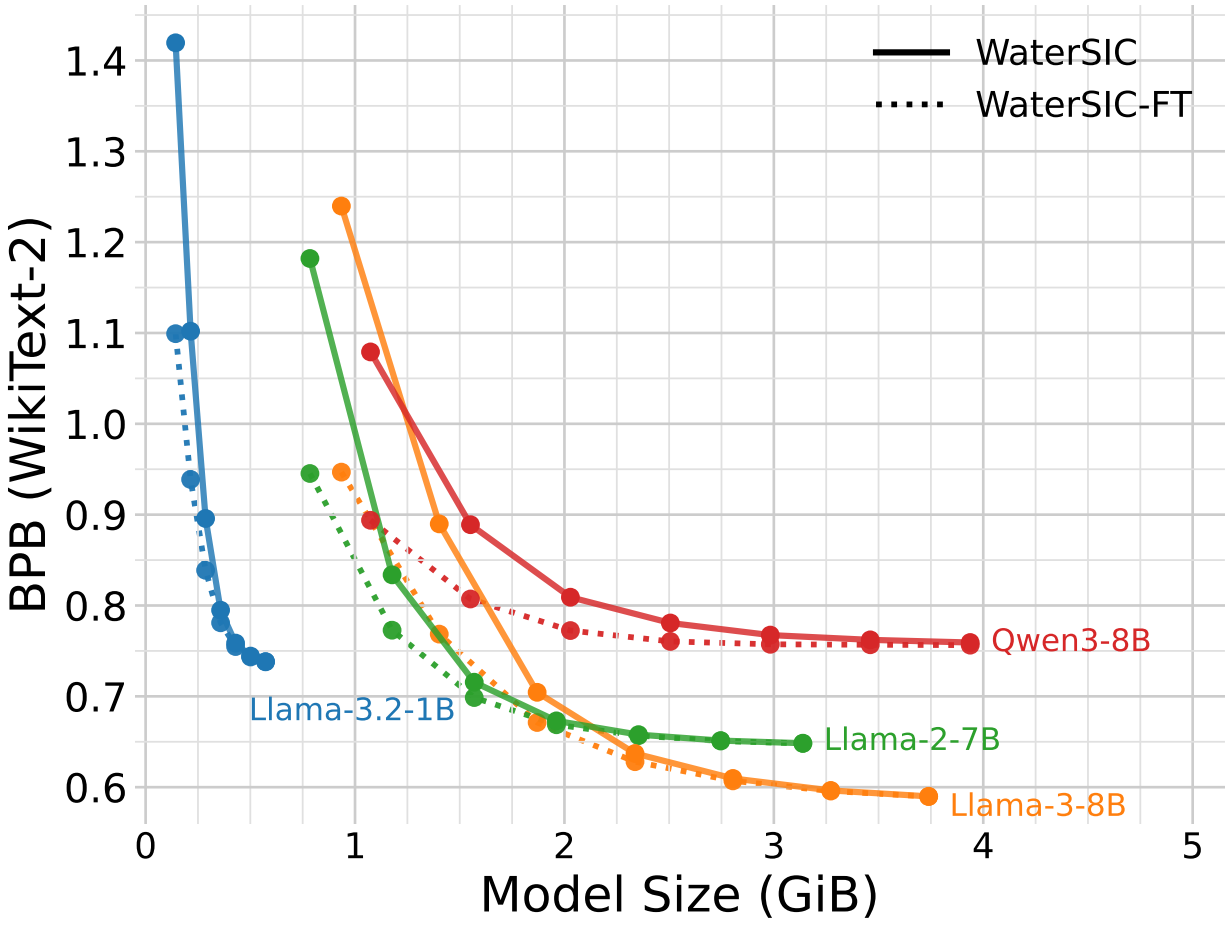

Figure 1.1: WikiText-2 bits-per-byte (BPB) vs. compressed model size (GiB) for WaterSIC and WaterSIC-FT across multiple base models. Lines connect the same model compressed with the same algorithm at various bit rates.

is the goal of this thesis.

1.2 Contributions

The central contribution of this thesis is *WaterSIC*, a layerwise weight-only quantizer that allocates different quantization rates to different columns (in-features) of the weight matrix according to the classical reverse-waterfilling rule. To the best of our knowledge, every previously published PTQ algorithm uses the same quantization rate for every column of W, in contradiction to the classical information-theoretic principle for parallel Gaussian sources. The diagonal entry ℓ_{ii} of the Cholesky factor of Σ_X has a natural interpretation as the conditional standard deviation of the i-th coordinate of X given the preceding coordinates, i.e., the root-mean-square of the linearly-unpredictable part of X_i; WaterSIC's central principle is to assign per-column rate inversely proportional to this linear unpredictability.

The technical contributions are as follows:

- An information-theoretic formulation of the layerwise weight-only quantization prob-

lem as rate-distortion coding of a Gaussian source with side information Σ_X, and a *waterfilling lower bound* on the rate-distortion function that holds uniformly over all input covariance matrices (Chapter 3).

- A proof that on iid Gaussian weights with input activation covariance Σ_X, WaterSIC achieves a rate within
$$\tfrac{1}{2}\log_2\left(\tfrac{2\pi e}{12}\right) \approx 0.255 \text{ bits} \tag{1.1}$$
of this lower bound, uniformly over Σ_X and at all rates. The bound is the standard high-rate gap between scalar uniform quantization and asymptotically optimal vector quantization, and our algorithm matches it.

- A proof that the popular GPTQ algorithm has an *arbitrarily large* rate gap to the same lower bound, depending on the conditioning of Σ_X, and that the gap of its entropy-coded variant Huffman-GPTQ is bounded but strictly larger than that of WaterSIC. A consequence of the theory worth flagging here is that WaterSIC's rate-distortion behavior depends only on $|\Sigma_X|$ and is therefore *invariant under rotations* $U\Sigma_X U^\top$, whereas Huffman-GPTQ's behavior is affected (typically degraded) by such rotations.

- A practical algorithm (Chapter 4) that combines the per-column rate allocation with a series of refinements (LMMSE shrinkage, activation drift correction, residual-stream compensation, diagonal rescalers, joint quantization of attention projections, and an adaptive mixing of unquantized and quantized covariances) required to deploy WaterSIC on real LLMs.

- A post-quantization finetuning variant, WaterSIC-FT, that adjusts only the per-row and per-column floating-point rescalers (two vectors totalling roughly 0.25% of model parameters) via KL distillation from the unquantized teacher, recovering substantial accuracy at very low bitrates.

- An empirical evaluation on the Llama and Qwen families across model sizes from 1B to 70B parameters and rates from 1 to 4 bits per weight (Chapter 5), establishing new state-of-the-art results on WikiText-2 perplexity throughout this range. On a suite of zero-shot accuracy benchmarks (ARC, HellaSwag, PIQA, SocialIQA, Winogrande, MMLU, OpenBookQA, SciQ, TriviaQA), WaterSIC matches or exceeds prior methods on the majority of tasks at every tested bitrate, with the largest gains concentrated in the aggressive low-bit regime.

Figure 1.1 previews the comparison.

1.3 The Anatomy of a PTQ Method

To position our contribution, it is useful to recall that a modern PTQ method consists of four largely-independent ingredients:

- The *base quantizer* takes a vector of high-resolution floating-point numbers and converts it to a lower-resolution version. Examples range from scalar uniform quantization (round

to a grid of stepsize Δ) to structured codebooks such as NVFP4 (16 floats $\to$ 16 four-bit codes plus an 8-bit microscaling coefficient), 8-dimensional vector codebooks such as E8P from QuIP# [7], learned vector codebooks such as AQLM [8], or trellis codes such as QTIP [9].

- The *scaling method* assigns scaling coefficients with various granularities so that the data fits within the dynamic range of the base quantizer. Common strategies are `absmax`, percentile-based scaling, and learned per-channel scaling [10–12]. This step is critical for handling weight outliers in models such as the OPT and early Llama families.

- *Calibration* adapts quantization error to the statistics of activations. Instead of applying the base quantizer uniformly across the matrix (round-to-nearest, RTN), calibration shapes the spectrum of the quantization error so that it lies in directions of low input variance, an idea originating in GPTQ [13] and extended by LDLQ [14].

- *Finetuning* adjusts continuous-valued parameters of the quantized model after the integer codes have been produced, often recovering further perplexity at low bitrates [8,15,16].

This thesis adopts the simplest possible setting in three of these four ingredients: a uniform base quantizer (rounding to a scalar grid), no learned scaling, and a lightweight finetuning step that adjusts only floating-point rescalers. The single ingredient where we depart from the literature is calibration: WaterSIC chooses the per-column quantization stepsize by waterfilling on the Cholesky diagonal of Σ_X, rather than using a single stepsize for every column. Reported gains are therefore additive with those of more sophisticated base quantizers, learned scaling, or stronger finetuning, and could be combined with them in principle.

1.4 Entropy Coding in Place of Range Constraints

In place of a hard scaling, our base quantizer is deliberately simple: $q_\epsilon(x)$ rounds each coordinate independently to an equispaced ϵ-grid. The output is a list of (possibly unbounded) integers. Instead of constraining the range of these integers via scaling, we compress them with a generic high-quality lossless coder (Lempel-Ziv, Zstandard, or Huffman), producing a finite but variable-length bitstring [1,17–19]; recent NVIDIA Blackwell hardware adds direct support for this style of weight encoding [20].[1] Tweaking ϵ smoothly adjusts the rate: when $\epsilon \to 0$ the entropy of the integers grows and the average compression rate increases. Entropy coding also automatically handles outliers: occasional large integers receive long bit-descriptions, but their infrequency means they barely affect the average rate.

This combination of integer rounding and entropy coding is what makes scalar quantization information-theoretically competitive, and is the reason the $\frac{1}{2}\log_2(2\pi e/12) \approx 0.255$ bit gap is the right target. A practical benefit is that *the quantization rate can be varied almost continuously* by tweaking ϵ, in contrast to vector codebooks whose rates are quantized to

[1] For example, the Llama-3 family is originally stored in BF16, but the entropy of its weights is only about 10.5 bits/weight, mostly concentrated in the mantissa bits.

$\log_2$ of a finite codebook size. WaterSIC's per-column rate allocation exploits this continuous tunability directly.

1.5 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 develops the technical background (the layerwise weight-only quantization problem, scalar quantization, rate-distortion of Gaussian sources, and reverse waterfilling) and surveys prior work on PTQ for LLMs. Chapter 3 contains the theoretical analysis: it formulates the layerwise problem as rate-distortion coding with side information Σ_X, derives the waterfilling lower bound, identifies GPTQ as successive-interference-cancellation decoding of a particular lattice, and proves the 0.255-bit guarantee for WaterSIC together with the unbounded gap for GPTQ. Chapter 4 describes the practical algorithm used to compress real LLMs: per-column refinements (LMMSE, drift, residual-stream, diagonal rescalers), joint quantization of attention projections, adaptive covariance mixing, entropy coding and implementation details, and the WaterSIC-FT finetuning variant. Chapter 5 reports empirical results across the Llama and Qwen families at 1 to 4 bits per weight, the effect of calibration and finetuning corpus, and discusses limitations and future work. Three appendices follow: hyperparameters (Appendix A), diagnostic plots and ablations (Appendix B), and zero-shot downstream-task evaluations on ARC, HellaSwag, PIQA, SocialIQA, Winogrande, MMLU, OpenBookQA, SciQ, and TriviaQA (Appendix C).

Bibliographic note. This thesis is based in substantial part on joint work with Semyon Savkin, Or Ordentlich, and Yury Polyanskiy that was accepted as a spotlight poster for ICML 2026 under the title "WaterSIC: information-theoretically (near) optimal linear layer quantization." The author of this thesis is the lead author of that publication and led the design of the algorithm, the implementation, and the experimental evaluation; the theoretical analysis in Chapter 3 is joint work with the coauthors, with O. Ordentlich and Y. Polyanskiy contributing the lattice-theoretic framework and S. Savkin contributing parts of the high-resolution analysis. The thesis expands the conference version with an extended background and related works section.

Chapter 2

Background and Related Work

This chapter has two purposes. Section 2.1 establishes the technical setting and the information-theoretic vocabulary that the rest of the thesis depends on: the layerwise weight-only quantization problem, the basic facts about scalar quantization, and the rate-distortion theory of Gaussian sources, including the reverse-waterfilling rule that underlies WaterSIC's per-column rate allocation. Section 2.2 surveys prior work on PTQ for LLMs, organized thematically rather than chronologically, and indicates the relationship of each thread to WaterSIC.

2.1 Technical Background

2.1.1 Linear Layers in LLMs and the Layerwise Quantization Problem

Modern transformer-based LLMs spend the overwhelming majority of their inference time and memory on dense linear layers, namely the projections inside multi-head attention (w_q, w_k, w_v, w_o) and the feed-forward expansions (w_{up}, w_{gate}, w_{down}). For a model with L transformer blocks of hidden dimension h and FFN intermediate dimension proportional to h, linear-layer weights account for the vast majority of total parameters, with the small remainder split between input/output embeddings and (very small) layer-norm parameters.

Each linear layer can be written as

$$Y = WX, \tag{2.1}$$

where $W \in \mathbb{R}^{a\times n}$ is a fixed weight matrix (n input features, a output features) and $X \in \mathbb{R}^n$ is an activation vector. Post-training quantization (PTQ) replaces W by a low-precision approximation $\hat{W}$ that can be stored using fewer bits, while keeping X in its original numerical format (BF16 or FP16). As discussed in Section 1.1, this *weight-only* regime is the dominant deployment regime for LLMs.

We focus on the *layerwise* formulation: given a single linear layer with weight matrix $W \in \mathbb{R}^{a\times n}$, a calibration distribution over $X \in \mathbb{R}^n$, and a target average bitrate R bits per weight, produce a quantized $\hat{W}$ that minimizes the expected mean-squared error of the layer

output:

$$D(\hat{W}) \stackrel{\text{def}}{=} \frac{1}{na}\,\mathbb{E}\big\|(W-\hat{W})X\big\|_2^2 \;=\; \frac{1}{na}\,\mathrm{tr}\big((W-\hat{W})\,\Sigma_X\,(W-\hat{W})^\top\big), \tag{2.2}$$

where $\Sigma_X = \mathbb{E}[XX^\top] \in \mathbb{R}^{n\times n}$ is the input activation covariance, often called the *Hessian* in the GPTQ literature and estimated from a calibration set in practice. The factor $1/(na)$ normalizes per weight entry; we interchangeably refer to D in absolute terms and as the *per-entry MSE*.

Several structural conventions are standard. First, under (2.2) the rows of W decouple, so weight-only quantization typically operates one row at a time. Second, codebooks are required to be *data-independent*: $\hat{W}$ must lie in some set $\mathcal{C}^a \subseteq (\mathbb{R}^n)^a$ that does not depend on Σ_X or on the realization of X. Only the bitstream describing the chosen codeword may depend on calibration. Third, in practice W is quantized one column at a time, with a scalar codebook applied to each column, possibly with column-dependent stepsize. Whether the same stepsize should be used for every column is one of the central questions of this thesis.

Despite a large literature of layerwise PTQ algorithms (some of which we survey in Section 2.2), the question of how close any of them comes to the information-theoretic limit on the rate-distortion tradeoff in (2.2) has not previously been addressed. Establishing that limit is the goal of Chapter 3; framing it informally is the goal of the rest of this section.

2.1.2 Quantization Schemes

This subsection collects standard facts about the quantizers used as building blocks throughout the thesis. A reader familiar with scalar quantization, per-channel scaling, and the distinction between PTQ and QAT can skip ahead to Section 2.1.3.

Scalar uniform quantization. A *scalar uniform quantizer* of stepsize $\Delta > 0$ rounds an input $w \in \mathbb{R}$ to the nearest point of the lattice $\Delta\mathbb{Z}$, producing an integer code $z = \mathrm{round}(w/\Delta)$ and reconstruction $\hat{w} = \Delta z$. At rate R bits per scalar (so the codebook covers 2^R levels), and assuming the input distribution is sufficiently smooth on this interval relative to Δ, the quantization error $\hat{w} - w$ is approximately uniform on $[-\Delta/2, \Delta/2)$ with variance $\Delta^2/12$. This is the building block of essentially every deployed weight-only PTQ algorithm, including GPTQ [13] and WaterSIC.

Symmetric vs. asymmetric quantization. Two minor variants differ in whether the codebook is symmetric around zero ($\hat{w} \in \{-(2^{R-1}-1), \dots, 2^{R-1}-1\} \cdot \Delta$) or shifted by a learned zero-point z_0 ($\hat{w} = (z - z_0)\Delta$). Symmetric quantization is preferred for weight quantization, since weight distributions are approximately symmetric around zero; asymmetric quantization is more common for KV-cache and activations, whose distributions are often shifted.

Granularity of scaling. The stepsize Δ may be assigned at several levels of granularity: *per-tensor* (a single Δ for the whole matrix), *per-channel* (one Δ per row or column), *per-group* (one Δ per group of g consecutive entries, typically $g \in \{32, 64, 128\}$), or *per-block* (e.g., the 16-element groups used by NVFP4 / MXFP4). Finer granularity reduces quantization

error but adds storage overhead for the stepsize metadata; common modern choices are per-channel or per-group (g=128) for weights and per-group for activations. Per-column rate assignment in WaterSIC can be thought of as the analog of per-channel scaling at the level of the rate budget rather than the dynamic range.

Non-uniform scalar codebooks. Beyond the uniform grid, several methods learn a scalar codebook of 2^R levels adapted to the weight distribution. Dettmers et al. [21] introduce NF4, a 4-bit codebook derived from quantiles of a standard normal; Elhoushi and Johnson [22] learn the 2^R levels directly. These methods are alternatives to the uniform quantizer in the *base quantizer* slot of the PTQ pipeline (Section 1.3) and are largely orthogonal to the calibration-level innovation of WaterSIC.

Vector quantization. Recent works replace the scalar codebook with a higher-dimensional vector codebook: QuIP# [7] uses an 8-dimensional lattice codebook (E8P); AQLM [8] learns a codebook with elements obtained as sums of trained codeword vectors; QTIP [9] uses a high-dimensional trellis code with a pseudorandom value function and fast streaming decoder; NestQuant [23] uses nested E_8 lattices with Voronoi-code addressing [4]. Vector quantizers recover most of the 0.255-bit gap between scalar uniform quantization and the rate-distortion limit, but at the cost of more elaborate encoding and decoding circuitry, with the cost differing substantially across methods: lattice-based codebooks such as E8P, QTIP, and NestQuant admit fast (often LUT-free) decoding, while learned codebooks such as AQLM require multiple codebook lookups and summations at every dequantization, and a beam-search encoding pass at quantization time. Methods that retain a scalar grid (uniform, NF4, WaterSIC) decode at the same cost as standard fp16 inference. We discuss vector quantizers further in Section 2.2.2.

PTQ vs. QAT. Post-training quantization runs the quantization algorithm on a frozen pretrained model using only a small calibration set ($\sim 10^3$ sequences), with no gradient steps to the model weights. *Quantization-aware training* (QAT) continues training the model with simulated quantization in the forward pass and straight-through estimators in the backward pass, typically requiring 10-20% of the original training data [15]. WaterSIC is a PTQ algorithm; WaterSIC-FT adds a lightweight finetuning step that adjusts only the per-row and per-column floating-point rescalers, leaving the integer codes frozen, and so sits well below QAT in compute cost.

2.1.3 An Information-Theoretic View

The information-theoretic view of weight-only PTQ treats W as an information source and $\hat{W}$ as a lossy code. The natural reference points are the rate-distortion function and the high-rate behavior of scalar quantizers.

Rate-distortion. For a real-valued source $W \sim p(w)$ and squared-error distortion, the rate-distortion function is

$$R^*(D) = \inf_{p(\hat{w}|w)\,:\,\mathbb{E}[(W-\hat{W})^2]\le D} I(W;\hat{W}), \tag{2.3}$$

giving the smallest mutual information at which W can be encoded with average squared error at most D. For a Gaussian source $W \sim \mathcal{N}(0, \sigma^2)$,

$$R^*(D) \;=\; \tfrac{1}{2}\log_2(\sigma^2/D), \qquad 0 < D \le \sigma^2. \tag{2.4}$$

This is the smallest rate, in bits per scalar, at which a Gaussian source can be encoded with mean-squared error at most D, achieved in the limit of long block lengths by random coding [24].

The 0.255-bit gap of scalar uniform quantization. A standard high-rate analysis [25] shows that as $D \to 0$, the rate achievable by scalar uniform quantization of a Gaussian source with subsequent entropy coding satisfies

$$R_{\text{scalar}}(D) \;=\; \tfrac{1}{2}\log_2\Big(\tfrac{2\pi e}{12}\cdot \sigma^2/D\Big) + o(1). \tag{2.5}$$

The factor $\frac{2\pi e}{12}$ is the ratio between the second moment of a uniform distribution on a unit interval and that of a Gaussian achieving the same differential entropy. It corresponds to a constant rate gap of

$$\tfrac{1}{2}\log_2(\tfrac{2\pi e}{12}) \;\approx\; 0.255 \text{ bits}. \tag{2.6}$$

This is the price of using scalar uniform quantization in place of asymptotically-optimal vector quantization. It is small enough that scalar-uniform-with-entropy-coding is the standard approach in modern LLM PTQ algorithms, and it is the gap that WaterSIC achieves uniformly over input covariances Σ_X in Theorem 3.3 of Chapter 3.

Parallel Gaussian sources and reverse waterfilling. When the source is a vector $W \in \mathbb{R}^n$ with independent Gaussian components $W_i \sim \mathcal{N}(0, \sigma_i^2)$, the rate-distortion function decomposes across components, but with the optimal per-component distortions allocated by the *reverse waterfilling* rule:

$$D_i \;=\; \min\{\sigma_i^2, \theta\}, \qquad R_i \;=\; \tfrac{1}{2}\log_2(\sigma_i^2/D_i)_+, \tag{2.7}$$

where θ is a parameter (the *water level*) chosen so that $\sum_i D_i = nD$. Components with $\sigma_i^2 \le \theta$ receive zero rate (they are described as zero); components with $\sigma_i^2 > \theta$ receive enough rate to bring their distortion down to θ. The aggregate rate is

$$R_{\text{wf}}(D) \;=\; \frac{1}{n}\sum_{i\,:\,\sigma_i^2>\theta} \tfrac{1}{2}\log_2(\sigma_i^2/\theta). \tag{2.8}$$

A nontrivial point: under squared-error distortion, the optimal allocation is *unequal* across components; components with larger variance receive more bits. A scheme that uses the same rate R for every component is provably suboptimal whenever the variances σ_i^2 are not all equal, with the gap to optimal growing as the spread of the spectrum $\{\sigma_i^2\}$ grows.

Connection to the layerwise PTQ problem. The objective in (2.2) is not an unweighted MSE on W; it is weighted by Σ_X. To bring the parallel-Gaussian picture to bear, factor $\Sigma_X = LL^\top$ (Cholesky decomposition, with L lower-triangular and diagonal entries $\ell_{ii} > 0$). Define the transformed weight $\tilde{W} = WL$ and reconstruction $\hat{\tilde{W}} = \hat{W}L$; then a short calculation gives

$$D(\hat{W}) \;=\; \frac{1}{na}\mathbb{E}\|\tilde{W} - \hat{\tilde{W}}\|_F^2, \tag{2.9}$$

i.e., the weighted layerwise objective is equivalent to an unweighted MSE on the transformed weight. The columns of $\tilde{W}$ inherit different scales: the i-th column has expected squared norm scaling with ℓ_{ii}^2. By the parallel-Gaussian analysis above, an information-theoretically optimal scheme should allocate *different rates* to different columns, with the allocation determined by the diagonal of L.

The diagonal element ℓ_{ii} has a natural interpretation worth pausing on: it is the conditional standard deviation of the i-th coordinate of X given the preceding $i-1$ coordinates, i.e., the root-mean-square of the residual after linear regression of X_i on $X_1, \dots, X_{i-1}$. Coordinates that are highly predictable from their predecessors have small ℓ_{ii}; coordinates carrying genuinely new "innovation" have large ℓ_{ii}. The per-column rate allocation should therefore be inversely proportional to the linear unpredictability of each input coordinate from the others: columns whose corresponding input feature is already well-explained by preceding features deserve coarser quantization. This is the principle that WaterSIC implements.

The conceptual claim of this thesis, made rigorous in Chapter 3, is that essentially all existing PTQ algorithms for LLMs, including GPTQ and the closest baseline Huffman-GPTQ, use the *same* quantization rate for every column, leaving a potentially large gap to the rate-distortion frontier. WaterSIC is the algorithm that closes this gap by allocating per-column rates according to the waterfilling rule.

Why iid Gaussian weights are a useful model. The theoretical analysis in Chapter 3 treats W as a matrix of iid $\mathcal{N}(0, \sigma_W^2)$ entries. Real LLM weights are of course not exactly Gaussian, but they are surprisingly close: on Llama-2-7B we measured the Kolmogorov-Smirnov distance from the empirical weight distribution to a maximum-likelihood Gaussian, and found mean $D_{\text{Gauss}} < 1\%$ for the feed-forward projections (w_1, w_2, w_3) and for w_v, w_o across all 32 transformer blocks, with w_q and w_k slightly worse and occasionally fit better by a Laplace distribution (full numbers in Appendix B). Two further considerations support the iid Gaussian model as a useful starting point. First, the Hadamard or random-rotation preprocessing used by QuIP [14], QuaRot [26], and SpinQuant [27] explicitly Gaussianizes the weight distribution before quantization: each post-rotation entry is a sum of $\Theta(n)$ original entries weighted by random signs, which is approximately Gaussian by the central limit theorem. Second, the weight outliers prominent in early LLM families (OPT, Bloom) appear to vanish in modern training pipelines with improved optimizers, layer-norm placements, and learning-rate schedules [28]. The iid Gaussian assumption is therefore best read as the asymptotic regime that modern training is moving toward, not as a literal description of every weight matrix.

Notation. Throughout the thesis we use uppercase letters for matrices (W, X, Y, Σ_X, L) and lowercase letters for vectors and scalars. The activation random vector $X \in \mathbb{R}^n$ has covariance $\Sigma_X = \mathbb{E}[XX^\top]$, factored throughout as $\Sigma_X = LL^\top$ with L lower-triangular and diagonal entries ℓ_{ii}. Rows of the weight matrix $W \in \mathbb{R}^{a\times n}$ are indexed $1, \dots, a$; columns $1, \dots, n$; the i-th column is denoted $W_{:,i}$. The expected layerwise distortion is denoted D, and the average bitrate R is in bits per weight entry. We write $\mathrm{tr}(M)$ for the trace and $|M|$ for the absolute value of the determinant of a square matrix M. The notation $(x)_+ = \max(x, 0)$ is used in waterfilling expressions.

2.2 Prior Work on PTQ

The literature on post-training quantization for LLMs is large and growing rapidly. We organize the most relevant work into five themes corresponding to the directions in which a quantization algorithm can be improved: the GPTQ algorithmic lineage, codebook and equivalent-transformation methods, loss-aware and activation-aware variants, variable-rate methods together with the information-theoretic perspective they connect to, and post-quantization finetuning together with the broader Pareto-frontier question. For each theme we sketch the central idea, point to representative works, and indicate the relationship to WaterSIC.

2.2.1 The GPTQ Lineage and Spectrum-Shaping Quantization

The naive baseline for layerwise weight-only quantization is *round-to-nearest* (RTN), which independently rounds each weight to the closest grid point. RTN ignores the structure of Σ_X, and at low rates its output is poor.

GPTQ [13] substantially improves on RTN by quantizing W column by column, using a closed-form correction that propagates the rounding error of already-quantized columns into the input of subsequent columns. The result is provably equivalent to LDLQ from QuIP [14]: given the Cholesky factorization $\Sigma_X = LL^\top$, the algorithm uses L to "shape the spectrum" of the quantization error so that it lies in the directions of low input variance. Concretely, the input to the quantizer for the i-th column is $W_{:,i}$ adjusted by a linear combination of the previous columns' quantization errors, with coefficients drawn from L. Modeling the per-coordinate quantization error as additive noise of variance $\Delta^2/12$ (the second moment of a uniform distribution on a stepsize-Δ cell), this yields a closed-form expression for the layerwise distortion as a function of Δ and L [14,29]. Birnick [29] explicitly identify GPTQ with successive-interference-cancellation (SIC) decoding of a particular lattice; this is the lattice-quantization viewpoint adopted in Chapter 3.

Two limitations of GPTQ-as-stated motivate the rest of the literature, including this thesis. First, GPTQ uses the *same* stepsize Δ for every column. As Chapter 3 shows, this leaves an arbitrarily large rate gap to the information-theoretic lower bound when the diagonal of L is non-uniform. Second, GPTQ as originally formulated does not entropy-code the resulting integer codes; under the more reasonable assumption that the encoder describes them with entropy coding, the rate-distortion behavior changes substantially. Huffman-GPTQ (HPTQ), introduced by Chen et al. [1], is the entropy-coded version of GPTQ and is the headline

baseline against which we report most of our results.

A growing body of work refines the GPTQ recipe in directions adjacent to ours. Qronos [30,31] and the equivalent QA-LDLQ correction independently derived in NestQuant [23] and developed in detail by Savkin [4, Ch. 5] both extend LDLQ to account for the fact that earlier layers have already been quantized when later layers are processed: the input activation to the present layer is $\hat{X} \neq X$, and a closed-form modification to the layerwise objective compensates for the resulting drift. We adopt this correction as one of WaterSIC's modular components (see Sections 4.1 and 2.2.3 below). NestQuant [23] additionally replaces the scalar lattice with a structured higher-dimensional lattice (the Gosset lattice E_8 via Voronoi codes), recovering some of the $\frac{1}{2}\log_2(2\pi e/12)$-bit gap of scalar uniform quantization at the cost of a higher-dimensional codebook. WaterSIC takes the opposite tack: it keeps the scalar lattice (and the 0.255-bit gap) but matches the IT lower bound up to that gap by varying the per-column rate allocation.

2.2.2 Codebooks and Equivalent Transformations

Codebook and vector quantization methods. A second line of work improves the *base quantizer* component by replacing the scalar uniform grid with a more sophisticated codebook. Elhoushi and Johnson [22] optimize a scalar codebook for 4-bit quantization. QuIP# [7] proposes an 8-dimensional vector codebook E8P of size 2^{16} with a fast decoding algorithm. AQLM [8] uses a vector codebook with elements obtained as sums of learned vectors; its encoding alternates beam search with codebook least-squares updates, and its decoding requires summing multiple codebook entries per dequantization. QTIP [9] uses a high-dimensional trellis code in which the trellis transition is a bit shift and node values are computed by a pseudorandom function of the node index, avoiding any explicit storage of the trellis and enabling fast streaming decoding. NestQuant [23] uses lattice-based encoding without lookup tables.

These methods are largely *orthogonal* to WaterSIC. WaterSIC determines *how much rate* to allocate to each column of the weight matrix; the codebook used to spend that rate within a column is a separate design choice. In our experiments we use a scalar uniform codebook, but combining WaterSIC with a structured higher-dimensional codebook from the works above is a natural extension.

Equivalent-transformation methods. Applying equivalent linear transformations to weights and activations can improve a downstream quantizer by reshaping the source distribution. QuIP [14] rotates weights with random Hadamard matrices, which incoherently spreads any concentration in the original weight distribution. QuaRot [26] applies a similar technique for activation quantization. FlatQuant [32] and SpinQuant [27] use an optimizer to find a best-performing transformation.

Per-channel scaling for outlier handling. A related family of methods exploits the identity $XW^\top = (X\,\mathrm{diag}(s)^{-1})(W\,\mathrm{diag}(s))^\top$ to redistribute dynamic range between weights and activations on a per-input-channel basis. SmoothQuant [33] picks the scales s_i to balance the per-channel ranges of X and W, making both matrices similarly easy to quantize.

AWQ [12] targets weight-only quantization and uses the same identity to scale up the channels associated with large-magnitude input activations, on the grounds that errors in those channels matter more for the layer output; the scales are chosen as $s_i = b_i^\alpha$ where b_i is the average activation magnitude in channel i and α is selected by grid search. AWQ is one of the strongest published weight-only baselines at low bitrates and is included throughout the evaluation in Chapter 5. Per-channel scaling is closely related to per-channel rate allocation: both exploit a per-input-channel degree of freedom, but along orthogonal axes (dynamic range versus rate budget), so the two ideas are in principle composable.

These methods modify the input to the quantizer; WaterSIC modifies the rate allocation *within* the quantizer. They are orthogonal: a Hadamard rotation followed by WaterSIC, or per-channel SmoothQuant scaling followed by WaterSIC, are both compatible compositions and would be natural to investigate.

2.2.3 Loss-Aware and Activation-Aware Methods

Loss-aware methods. GPTQ and its variants minimize layerwise output MSE. A line of work argues that this is a proxy for the quantity that matters (the model's perplexity or KL divergence to the unquantized model), and incorporates a Fisher-information-based objective instead. Storing the full Fisher information matrix is infeasible since it is quadratic in the weight matrix size, so practical methods rely on structured approximations. YAQA [3] decomposes the Fisher matrix into a Kronecker product of two matrices corresponding to pairs of rows and pairs of columns. GuidedQuant [34] assumes that Fisher information coefficients corresponding to parameters in different output channels are zero, and applies per-group averaging.

Loss-aware modifications are complementary to WaterSIC's rate-allocation contribution: WaterSIC determines a rate budget per column under MSE loss, but the same rate-allocation principle could be applied with a Fisher-weighted objective. We do not pursue this here; it is among the most natural future directions and is discussed in Section 5.3.

Activation-quantization-aware methods. Most weight-only PTQ algorithms calibrate against the *unquantized* input activations X, which is correct when activations remain in BF16/FP16 at deployment. When earlier layers have themselves been quantized (so the layer's actual input is some $\hat{X} = X + \delta$ with δ a quantization-noise term), direct calibration against X leads to systematic bias. Qronos [30,31] and the equivalent QA-LDLQ correction [4,23] address this by replacing the target weight W with a drift-corrected $WH(H + J)^{-1}$ (where $H = \Sigma_X$, $J = \mathbb{E}[\delta\delta^\top]$) and running the standard layerwise procedure on the modified target with covariance $H + J$. The two derivations are independent; we use the name Qronos when referring to the algorithm but cite both sources.

WaterSIC composes naturally with the Qronos correction (both are layerwise, both depend only on second-order activation statistics), and our best empirical results combine the two (Section 4.1). The information-theoretic question of what the joint rate-distortion frontier of *simultaneous* weight and activation quantization looks like is not addressed by either Qronos or WaterSIC, and is one of the open problems discussed in Section 5.3.

2.2.4 Variable-Rate Methods and Information-Theoretic Perspectives

Variable-rate methods. Allocating different rates to different columns or different blocks of the weight matrix is the central technical idea of WaterSIC, and there is a small but growing literature on variable-rate weight-only PTQ. GPTVQ [35] alternates between vector quantization and per-row scaling, effectively allowing a mild form of unequal rate across rows. Chen et al. [1] explore entropy-coded variants of GPTQ (Huffman-GPTQ and Huffman-RTN) that achieve continuous rate variation by tweaking the quantizer stepsize, but with the same stepsize across all columns. Both works approach the problem from a numerical-optimization standpoint rather than from the rate-distortion perspective taken here.

What distinguishes WaterSIC is that the per-column rate allocation is not learned by gradient descent or chosen by heuristic, but is given in closed form by the classical reverse-waterfilling rule, applied to the Cholesky diagonal of Σ_X. Chapter 3 shows that this allocation is provably within 0.255 bits of the information-theoretic lower bound, uniformly over Σ_X. To our knowledge, this is the first rate-allocation strategy in the LLM PTQ literature with such a guarantee.

Information-theoretic perspectives. Several works approach LLM quantization from an information-theoretic angle, and the present thesis builds directly on this lineage. The classical reference for the high-rate behavior of scalar entropy-coded quantizers is Linder and Zamir [25], which establishes the $\frac{1}{2}\log_2(2\pi e/12) \approx 0.255$-bit gap appearing throughout this thesis. Birnick [29] interpret GPTQ as successive-interference-cancellation decoding of a specific lattice, an interpretation we adopt and extend in Chapter 3. Chen et al. [1] derive the high-rate rate-distortion behavior of Huffman-GPTQ and identify per-column rate allocation as the natural next step, the gap that this thesis closes. Savkin et al. [23] construct nested-lattice quantizers that approach the matrix-product rate-distortion limit of Ordentlich and Polyanskiy [36], providing a complementary information-theoretic baseline against which WaterSIC is benchmarked on Llama models in Chapter 5.

2.2.5 Post-Quantization Finetuning and the Pareto Frontier

Post-quantization finetuning. After PTQ has produced an initial $\hat{W}$, finetuning can recover further quality, particularly at low bitrates. The most aggressive method is full quantization-aware training (QAT) on the quantized model, which Liu et al. [15] reports works best with around 10-20% of available data. A less expensive approach is to quantize a layer and then fine-tune subsequent unquantized layers to compensate [16]. The cheapest approach finetunes only those parameters that remain in floating point (scales, codebooks, and rescalers) [7,8]. PV-tuning [2] sits between these regimes: it alternates updates to the continuous parameters with discrete updates to the integer codes themselves, which delivers strong results at very low bitrates but is correspondingly more expensive than continuous-only finetuning. WaterSIC-FT (Section 4.5) belongs to the cheapest category: we finetune only the per-row and per-column rescalers, and the integer codes are kept fixed.

Pareto frontier and open questions. A central open question in this area is the Pareto frontier of LLM quality versus the total number of bits used to store weights. Kim et al. [37] estimates the frontier based on weight and KV-cache compression by GPTQ, but newer algorithms achieve substantially better quality at the same bit budget. ParetoQ [15] estimates the Pareto frontier at below 2 bits per parameter, while Chen et al. [1] suggest slightly above 3 bits. Memorization studies are similarly inconclusive, with Allen-Zhu and Li [38] estimating 2.2 bits per parameter and Morris et al. [39] estimating 3.6. From an information-theoretic point of view, narrowing these estimates requires a quantization algorithm with a known gap to the rate-distortion frontier, which is precisely what WaterSIC provides.

Chapter 3

Theory: WaterSIC and GPTQ

This chapter develops the information-theoretic theory underlying WaterSIC. We formalize the layerwise weight-only quantization problem and define its rate-distortion function $D^*(R, \Sigma_X)$, derive a waterfilling lower bound by considering an easier problem in which the decoder is allowed to depend on Σ_X, and introduce the conceptual algorithm PlainWaterSIC together with the lattice/SIC view of GPTQ. The main result, proved in Section 3.4, is that WaterSIC's rate gap to the lower bound is at most 0.255 bits uniformly over Σ_X in the high-resolution limit, while the gap of GPTQ can be made arbitrarily large.

3.1 Setup and Fundamental Limits

Let us restrict attention to a particular linear layer in the network with weight matrix $W \in \mathbb{R}^{a\times n}$. The linear layer operates by taking (a vector of) input activations $X \in \mathbb{R}^n$ and producing $Y = WX$. The task of quantization is to replace W with quantized version $\hat{W}$, which can be represented by naR bits, aiming at minimizing the expected distortion

$$\begin{aligned} D &= \frac{1}{na}\mathbb{E}\|(W-\hat{W})X\|_F^2 \\ &= \frac{1}{na}\operatorname{tr}(W-\hat{W})\Sigma_X(W-\hat{W})^\top, \end{aligned} \tag{3.1}$$

where the expectation is with respect to the random variable X and $\Sigma_X = \mathbb{E}[XX^\top]$. In practice, the matrix Σ_X is estimated from calibration data.

For the analysis we model the rows of W as iid $\mathcal{N}(0, \sigma_W^2 I)$ random vectors in $\mathbb{R}^{1\times n}$. While this is certainly not a precise model, it lets us derive the fundamental limits of the weight-only quantization problem and design practical schemes that perform quite close to them. Furthermore, under this assumption, the rows of W can be quantized independently without loss of optimality (assuming n is large enough). We will therefore assume that $a = 1$ for the derivation of the theoretical bounds; this is without loss of generality for n large enough.

The IT limit in this case is defined as follows. Given $n > 0$, $\sigma_W^2 > 0$ and PSD matrix $\Sigma_X \in \mathbb{R}^{n\times n}$, an (R, D, Σ_X) weight-only quantization scheme consists of an encoder $f : \mathbb{R}^n \times \mathbb{R}^{n\times n} \to \{0,1\}^*$ and decoder $g : \{0,1\}^* \to \mathbb{R}^n$, where distortion D is given by (3.1)

(with $a = 1$) and rate R is defined as

$$R = \frac{1}{n}\mathbb{E}[\ell(f(W, \Sigma_X)]\,,$$

where $\ell(\cdot)$ is the length of the binary string $(\cdot)$. The fundamental limits are defined as

$$R^*(D, \Sigma_X) = \inf\{R \ : \ \exists(R, D, \Sigma_X) - \text{scheme}\}\,,$$
$$D^*(R, \Sigma_X) = \inf\{D \ : \ \exists(R, D, \Sigma_X) - \text{scheme}\}\,.$$

The main point of this section is to show that as $n \to \infty$ we have

$$D^*(R, \Sigma_X) = \sigma_W^2 |\Sigma_X|^{1/n} 2^{-2R+o(1)}\,,$$

where $o(1)$ is with respect to $R \to \infty$ and $|\Sigma_X|$ denotes determinant of a matrix. The standard entropy-coded GPTQ (a.k.a. Huffman-GPTQ) achieves

$$D^*_{\text{GPTQ}}(R, \Sigma_X) = 2^{-2R+o(1)} \frac{2\pi e}{12} \frac{\sigma_W^2}{n} \sum_{i=1}^{n} (\ell_{ii})^2\,,$$

where ℓ_{ii} is the i-th diagonal element of a lower triangular matrix of the Cholesky decomposition $\Sigma_X = LL^\top$. The newly proposed WaterSIC, though, achieves

$$D^*_{\text{WaterSIC}}(R, \Sigma_X) = 2^{-2R+o(1)} \frac{2\pi e}{12} \sigma_W^2 \prod_{i=1}^{n} (\ell_{ii})^{\frac{2}{n}}\,,$$

which (by AMGM) is always better than Huffman-GPTQ. However, what is more exciting is that $\prod \ell_{ii} = |L| = \sqrt{|\Sigma_X|}$ and therefore, D^*_{WaterSIC} is only a factor $\frac{2\pi e}{12}$ worse than IT limit $D^*(R, \Sigma_X)$. Since $\frac{1}{2}\log_2 \frac{2\pi e}{12} = 0.255$ bit, the rate gap between WaterSIC and IT limit is at most 0.255 bit, as claimed. Another difference is that Huffman-GPTQ's performance is affected (often reduced) by applying a random rotation $U\Sigma_X U^\top$. But WaterSIC's performance is invariant to such rotations, since it only depends on $|\Sigma_X|$.

3.2 Waterfilling Lower Bound

In the problem setup above, the decoder cannot depend on Σ_X. To obtain a lower bound on the smallest distortion attained by any weight-only quantizer, we consider an easier setup where g can also depend on Σ_X. Any lower bound for this easier setup is clearly also a valid lower bound for our setup. Consider the singular value decomposition (SVD) $\Sigma_X = V\Lambda V^\top$. Define $\tilde{W} = WV\Lambda^{1/2}$ and $\hat{\tilde{W}} = \hat{W}(V\Lambda^{1/2})^{-1}$. With this notation, we have that

$$D = \mathbb{E}\|\tilde{W} - \hat{\tilde{W}}\|^2.$$

Thus, the problem reduces to rate R quantization of a Gaussian source with independent components $\tilde{W} \sim \mathcal{N}(0, \sigma_W^2\Lambda)$. It is well-known that the optimal rate-distortion tradeoff for

this problem is given by (reverse) *waterfilling solution*[1]

$$R_{\mathrm{WF}}(D,\Sigma_X) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\log\left(\max\left\{1,\frac{\sigma_W^2\lambda_i}{\tau}\right\}\right),$$
$$\text{where } D = \frac{1}{n}\sum_{i=1}^{n}\min\{\sigma_W^2\lambda_i,\tau\}, \tag{3.2}$$

where $\Lambda = \mathrm{diag}(\lambda_1,\dots,\lambda_n)$. It is easy to see that

$$\forall D < \min\{\sigma_W^2\lambda_i\} : R_{\mathrm{WF}}(D,\Sigma_X) = \frac{1}{2}\log\left(\frac{\sigma_W^2|\Sigma_X|^{\frac{1}{n}}}{D}\right)$$
$$\triangleq R_{\mathrm{High-Rate}}(D,\Sigma_X). \tag{3.3}$$

To summarize, we have

Proposition 3.1.

$$\forall \Sigma_X \ : \ R^*(D,\Sigma_X) \geq R_{\mathrm{WF}}(D,\Sigma_X).$$

We stress that the main reason it is not possible to achieve this bound in practice is because decoder needs to able to operate in the PCA basis of Σ_X, but informing decoder of the SVD basis requires sending an $n \times n$ orthogonal matrix, which is of the same order of magnitude as the original matrix W.

3.3 GPTQ and PlainWaterSIC

Applying Cholesky decomposition, we can write $\Sigma_X = LL^\top$, where $L \in \mathbb{R}^{n\times n}$ is a lower triangular matrix. Our objective function can therefore be written as

$$D = \frac{1}{n}\mathbb{E}\|Y - \hat{W}L\|^2, \quad Y = WL. \tag{3.4}$$

Assume that the reconstruction $\hat{W}$ must belong to the scaled integer lattice $\alpha\mathbb{Z}^{1\times n}$. Given $y \in \mathbb{R}^{1\times n}$, the problem of finding $\hat{w} \in \alpha\mathbb{Z}^{1\times n}$ that minimizes $\|y - \hat{w}L\|^2$ is that of finding the closest vector to y in the lattice $\alpha\mathbb{Z}^{1\times n}L$. This problem is known to be NP-hard. Thus, one must resort to sub-optimal algorithms. One such option is successive interference cancellation (SIC). This corresponds to utilizing the lower triangular structure of L and sequentially deciding on the elements of $\hat{w}$ starting from $\hat{w}_n$ until reaching $\hat{w}_1$. Once we decide on the value of $\hat{w}_i$, we subtract its contribution (interference) to coordinates of y that were not used for quantization yet. This procedure is described in Algorithm 1 which we refer to as the ZSIC algorithm. In ZSIC, we do not restrict ourselves to reconstruction in $\alpha\mathbb{Z}^n$, but instead allow the reconstruction to be in $\mathbb{Z}^{1\times n}\mathcal{A}$, where $\mathcal{A}$ is a diagonal matrix with $|\mathcal{A}|^{1/n} = \alpha$. This lattice is a cartesian product $\prod_{i=1}^{n}(\alpha_i\mathbb{Z})$, whose point density is α^{-n} just like the lattice $\alpha\mathbb{Z}^{1\times n}$. However, a judicious choice of $\mathcal{A}$ optimizes the spacings as a function of L. The benefits of this optimization will immediately become clear. Note further that Algorithm 1 is

Algorithm 1 ZSIC

Inputs: $Y \in \mathbb{R}^{a\times n}$, lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and diagonal spacing matrix $\mathcal{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^{n\times n}$.

Outputs: $Z_{\mathrm{SIC}} \in \mathbb{Z}^{a\times n}$ ▷ $Z_{\mathrm{SIC}}\mathcal{A} \approx \mathrm{argmin}_Z \|Y - Z\mathcal{A}L\|_2^2$

$Z_{\mathrm{SIC}} \leftarrow 0^{a\times n}$ ▷ Initialize Z_{SIC} with zeros

for $i = n : 1$ **do**

$Z_{\mathrm{SIC},:,i} \leftarrow \mathrm{round}\left(\frac{Y_{:,i}}{\alpha_i \ell_{ii}}\right)$ ▷ $Y_{:,i}$ and $Z_{\mathrm{SIC},:,i}$ is the ith column of Y and Z_{SIC}, respectively

$Y \leftarrow Y - \alpha_i Z_{\mathrm{SIC},:,i} \cdot L_{i,:}$ ▷ $L_{i,:}$ is the ith row of L

end for

described for a matrix input, rather than a row. However, the algorithm simply applies the same quantization procedure to each one of the a rows of $Y = WL$.

Denote $\mathrm{CUBE} = \left[-\frac{1}{2}, \frac{1}{2}\right)^{1\times n}$. The following lemma characterizes the output of the ZSIC algorithm.

Lemma 3.2. *Assume we apply the ZSIC Algorithm 1 with $y \in \mathbb{R}^{1\times n}$, lower triangular $L \in \mathbb{R}^{n\times n}$, and diagonal matrix $\mathcal{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^n$. Then*

$$e_{\mathrm{SIC}} = y - z_{\mathrm{SIC}} \cdot \mathcal{A}L \in \mathrm{CUBE} \cdot \mathcal{A}\,\mathrm{diag}(L).$$

Proof of Theorem 3.2. Let $z_{\mathrm{SIC}}(y, L, \mathcal{A})$ be the result of applying the SIC algorithm with inputs $y \in \mathbb{R}^{1\times n}$ and $L, \mathcal{A} \in \mathbb{R}^{n\times n}$. The lemma follows from combining the two following observations:

1. The region of all $y \in \mathbb{R}^{1\times n}$ that are mapped to 0 is

$$\left\{y \in \mathbb{R}^{1\times n} : z_{\mathrm{SIC}}(y, L, \mathcal{A}) = 0\right\} = \mathrm{CUBE} \cdot \mathcal{A}\,\mathrm{diag}(L)$$

2. For any $z \in \mathbb{Z}^{1\times n}$ it holds that

$$z_{\mathrm{SIC}}(y + z\mathcal{A}L, L, \mathcal{A}) = z\mathcal{A} + z_{\mathrm{SIC}}(y, L, \mathcal{A}).$$

Thus, the ZSIC algorithm induces the partition of space to decision regions

$$\mathcal{D}_z = z\mathcal{A}L + \mathrm{CUBE} \cdot \mathcal{A}\,\mathrm{diag}(L), \ \forall z \in \mathbb{Z}^{1\times n} \tag{3.5}$$

and $z_{\mathrm{SIC}}(y, L, \mathcal{A}) = z\mathcal{A}$ iff $y \in \mathcal{D}_z$. Consequently, $e_{\mathrm{SIC}} = y - z_{\mathrm{SIC}}(y, L, \mathcal{A}) \in \mathrm{CUBE} \cdot \mathcal{A}\,\mathrm{diag}(L)$. □

Recall that in our setup the input to the ZSIC algorithm is $Y = WL$, where $W \sim \mathcal{N}(0, \sigma_W^2)$. It therefore follows that Z_{SIC} is a random vector supported on $\mathbb{Z}^{1\times n}$ and e_{SIC} is a random vector supported on $\mathrm{CUBE} \cdot \mathcal{A}\,\mathrm{diag}(L)$. As the ratio $\sigma_W / \det|\mathcal{A}|$ grows, the distribution

[1] The standard setup is for fixed-rate quantizers, but the converse holds also for variable-rate quantizers.

of e_{SIC} approaches the uniform distribution on $\text{CUBE} \cdot \mathcal{A}\,\text{diag}(L)$ (see Section 3.4) and we obtain

$$D_{\text{SIC}} \approx \frac{1}{n}\frac{1}{12}\sum_{i=1}^{n}(\alpha_i \ell_{ii})^2 \tag{3.6}$$

$$= \frac{|\mathcal{A}L|^{2/n}}{12}\frac{\frac{1}{n}\sum_{i=1}^{n}(\alpha_i \ell_{ii})^2}{\prod_{i=1}^{n}(\alpha_i \ell_{ii})^{2/n}} \tag{3.7}$$

$$= |\mathcal{A}|^{2/n}\frac{|\Sigma_X|^{1/n}}{12}\frac{\frac{1}{n}\sum_{i=1}^{n}(\alpha_i \ell_{ii})^2}{\prod_{i=1}^{n}(\alpha_i \ell_{ii})^{2/n}} \tag{3.8}$$

Denote by $H(\cdot)$ the Shannon entropy of a discrete random variable and by $h(\cdot)$ the differential entropy of a continuous random variable. As $\sigma_W/\det|\mathcal{A}|$ grows, we also have that

$$\sum_{i=1}^{n} H(Z_{\text{SIC},i}) \approx nh(W) - \log|\mathcal{A}| \tag{3.9}$$

$$= \frac{n}{2}\log\left(\frac{2\pi e \sigma_W^2}{|\mathcal{A}|^{2/n}}\right) \tag{3.10}$$

$$\approx \frac{n}{2}\log\left(\frac{|\Sigma_X|^{1/n}\sigma_W^2}{D_{\text{SIC}}}\frac{2\pi e}{12}\frac{\frac{1}{n}\sum_{i=1}^{n}(\alpha_i \ell_{ii})^2}{\prod_{i=1}^{n}(\alpha_i \ell_{ii})^{2/n}}\right), \tag{3.11}$$

where (3.9) is derived in Section 3.4, and in (3.11) we have used (3.8).

If we apply the ZSIC algorithm on a matrix $Y = WL$, where the rows of W are independently drawn from $\mathcal{N}(0, \sigma_W^2 I)$, each column $Z_{\text{SIC},:,i}$ of the algorithm's output will consist of iid entries from a distribution on $\mathbb{Z}$ with entropy $H(Z_{\text{SIC},i})$. Thus, if the number of rows a is sufficiently large, encoding $Z_{\text{SIC},:,i}$ to bits can be done via any standard entropy coding algorithm, and the expected number of bits in this description will be $aH(Z_{\text{SIC},i})(1+o(1))$, where $o(1)$ vanishes as $a \to \infty$. From the entropy coded binary descriptions of all n columns, the decoder can recover Z_{SIC} and set $\hat{W} = Z_{\text{SIC}}\mathcal{A}$. If the matrix $\mathcal{A}$ is determined by the encoder, it also needs to describe the entries $\alpha_1, \dots \alpha_n$ in bits. Since those are only n scalars, the cost of their description is negligible for $a \gg 1$.

It therefore follows that applying ZSIC algorithm with matrix $\mathcal{A} = \text{diag}(\alpha_1, \dots, \alpha_n)$ and describing each column of the resulting matrix Z_{SIC} using entropy coding approximately achieves the following tradeoff between rate and distortion

$$R_{\text{SIC}}(D, \Sigma_X) \approx R_{\text{High-Rate}}(D, \Sigma_X)$$
$$+ \frac{1}{2}\log\left(\frac{2\pi e}{12}\right) + \frac{1}{2}\log\left(\frac{\frac{1}{n}\sum_{i=1}^{n}(\alpha_i \ell_{ii})^2}{\prod_{i=1}^{n}(\alpha_i \ell_{ii})^{2/n}}\right), \tag{3.12}$$

for D small enough.

The canonical GPTQ algorithm is completely equivalent to the ZSIC algorithm with $\mathcal{A} = \alpha I$ [1,29]. When the matrix Z_{SIC} is further described using entropy coding, we get the *Huffman-GPTQ* algorithm explicitly proposed in [1] under the name *HPTQ*, which we use interchangeably with Huffman-GPTQ.

Algorithm 2 PlainWaterSIC weight-only quantization

Inputs: $W \in \mathbb{R}^{a\times n}$, PSD matrix Σ_X and point density $\alpha > 0$.
Outputs: $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}_+^n$, and binary vectors $B_1, \dots, B_n \in \{0,1\}^*$ encoding the n columns of Z_{SIC} s.t. $\widehat{W} = Z_{\text{SIC}} \operatorname{diag}(\alpha_1, \dots, \alpha_n)$.

Compute lower-triangular $L \in \mathbb{R}^{n\times n}$ such that $\Sigma_X = LL^\top$ ▷ Using the Cholesky decomposition
$\alpha_i \leftarrow \alpha \cdot \frac{|L|^{\frac{1}{n}}}{|\ell_{ii}|}, \quad \forall i \in [n]$ ▷ ℓ_{ii} is the ith diagonal element of L
$\mathcal{A} \leftarrow \operatorname{diag}(\alpha_1, \dots, \alpha_n)$
$Z_{\text{SIC}} \leftarrow \text{ZSIC}(WL, L, \mathcal{A})$ ▷ Apply Algorithm 1 with arguments $Y = WL$, L and $\mathcal{A}$
for $i = 1 : n$ **do**
  $B_i \leftarrow \text{EC}\left(Z_{\text{SIC},:,i}\right)$ ▷ Entropy coding for the ith column of $Z_{\text{SIC},:,i}$
end for

The last term in (3.12) is non-negative due to the arithmetic mean-geometric Mean (AMGM) inequality. It therefore follows that the choice $\mathcal{A} = \alpha I$ is sub-optimal in general, and the optimal choice (under the approximations leading to (3.12)) is

$$\alpha_i = \frac{c}{|\ell_{ii}|}, \quad i = 1, \dots, n, \tag{3.13}$$

where $c > 0$ is a constant that determines the density of the lattice $\mathbb{Z}^{1\times n}\mathcal{A}$. In particular, if we want the density to be α^{-n}, as for the lattice $\alpha\mathbb{Z}^{1\times n}$, we choose $c = \alpha \cdot |L|^{1/n}$.

We refer to the resulting algorithm for weight-only quantization that utilizes ZSIC together with this choice of $\mathcal{A}$ and entropy coding as PlainWaterSIC, and it is described in Algorithm 2. In the next section we describe several improvements of this algorithm, that result in the full WaterSIC, see Algorithm 3. Note that if we modify the computation of α_i in Algorithm 2 to $\alpha_i = \alpha \forall i = 1, \dots, n$, we get the HPTQ algorithm from [1].

The expression (3.12) for the rate-distortion tradeoff of ZSIC relied on approximations. The following theorem shows that these approximations become exact in the limit of high-rate/low-distortion; the proof is given in Section 3.4 below.

Theorem 3.3. *For any positive semi definite matrix* Σ_X

$$\begin{aligned}&\lim_{a\to\infty}\lim_{D\to 0} R_{\text{GPTQ}}(D, \Sigma_X) - R_{\text{WF}}(D, \Sigma_X)\\ &= \frac{1}{2}\log\left(\frac{2\pi e}{12}\right) + \frac{1}{2}\log\left(\frac{\frac{1}{n}\sum_{i=1}^n (\ell_{ii})^2}{\prod_{i=1}^n (\ell_{ii})^{2/n}}\right)\end{aligned} \tag{3.14}$$

and

$$\begin{aligned}&\lim_{a\to\infty}\lim_{D\to 0} R_{\text{WaterSIC}}(D, \Sigma_X) - R_{\text{WF}}(D, \Sigma_X)\\ &= \frac{1}{2}\log\left(\frac{2\pi e}{12}\right)\end{aligned} \tag{3.15}$$

3.4 Proof of the High-Resolution Limit

This section proves Theorem 3.3 from Section 3.1, restated here for convenience.

Theorem 3.4 (Theorem 3.3, restated)**.** *For any positive semi-definite matrix* Σ_X*,*

$$\lim_{a\to\infty}\lim_{D\to 0} R_{\mathrm{GPTQ}}(D,\Sigma_X) - R_{\mathrm{WF}}(D,\Sigma_X) = \tfrac{1}{2}\log\left(\tfrac{2\pi e}{12}\right) + \tfrac{1}{2}\log\left(\frac{\frac{1}{n}\sum_{i=1}^n \ell_{ii}^2}{\prod_{i=1}^n \ell_{ii}^{2/n}}\right),$$

$$\lim_{a\to\infty}\lim_{D\to 0} R_{\mathrm{WaterSIC}}(D,\Sigma_X) - R_{\mathrm{WF}}(D,\Sigma_X) = \tfrac{1}{2}\log\left(\tfrac{2\pi e}{12}\right).$$

The proof proceeds in two steps. We first develop two preliminaries: the lattice flatness factor, which controls how close the quantization error distribution is to uniform on the Voronoi region, and a corresponding statistical claim about the quantizer's input that becomes exact in the high-rate limit. We then assemble these into a joint analysis of Huffman-GPTQ and PlainWaterSIC.

Preliminaries: the flatness factor of a lattice. For a lattice $\Lambda \subset \mathbb{R}^{1\times n}$ and $\sigma > 0$, define the Λ-periodic function $f_{\sigma,\Lambda} : \mathbb{R}^{1\times n} \to \mathbb{R}$

$$f_{\sigma,\Lambda}(x) = \sum_{\lambda\in\Lambda} \phi_\sigma(x-\lambda), \tag{3.16}$$

where

$$\phi_\sigma(x) = (2\pi\sigma^2)^{-n/2} e^{-\frac{\|x\|^2}{2\sigma^2}} \tag{3.17}$$

The flatness factor of a lattice Λ is defined as [40,41]

$$\epsilon_\Lambda(\sigma) = \sup_{x\in\mathbb{R}^{1\times n}} \left| \frac{f_{\sigma,\Lambda}(x)}{1/\mathrm{covol}(\Lambda)} - 1 \right|. \tag{3.18}$$

It is well-known, see e.g. [41, Corollary 1] that

$$\epsilon_\Lambda(\sigma) = \sum_{\lambda'\in\Lambda^*\setminus\{0\}} e^{-2\pi^2\sigma^2\|\lambda'\|^2}, \tag{3.19}$$

where Λ^* is the dual lattice of Λ. Note that the dual lattice of $\mathbb{Z}^{1\times n}$ is $\mathbb{Z}^{1\times n}$. Furthermore, if $\tilde{\mathcal{A}} = \mathrm{diag}(\tilde{\alpha}_1,\dots,\tilde{\alpha}_n)$ for fixed $\tilde{\alpha}_1,\dots,\tilde{\alpha}_n > 0$, and $\tilde{\Lambda} = \alpha\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}$ for some $\alpha > 0$, then $\tilde{\Lambda}^* = \alpha^{-1}\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}^{-1}$ and

$$\epsilon_{\alpha\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}}(\sigma) = \sum_{z\in\mathbb{Z}^n\setminus\{0\}} \exp\left(-\left(\frac{\sigma}{\alpha}\right)^2 \cdot 2\pi^2\|z\tilde{\mathcal{A}}^{-1}\|^2\right). \tag{3.20}$$

We therefore have that for any fixed $\sigma,\tilde{\alpha}_1,\dots,\tilde{\alpha}_n > 0$

$$\lim_{\alpha\to 0} \epsilon_{\alpha\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}}(\sigma) = 0. \tag{3.21}$$

Preliminaries: statistics of the quantizer's input. We prove the following.

Lemma 3.5. *Let $W \sim \mathcal{N}(0, \sigma^2 I)$ be a Gaussian vector in $\mathbb{R}^{1\times n}$, $L \in \mathbb{R}^{n\times n}$ be a lower triangular matrix, and $\tilde{\mathcal{A}} = \operatorname{diag}(\tilde{\alpha}_1, \dots, \tilde{\alpha}_n) \in \mathbb{R}_+^{n\times n}$. Assume we apply Algorithm 1 with $Y = WL$, L and $\mathcal{A} = \alpha\tilde{\mathcal{A}}$, for some $\alpha > 0$. Then*

$$Z_{\mathrm{SIC},1,k} = \operatorname{round}\left(\frac{W_k}{\alpha\tilde{\alpha}_k} + e_k\right), \tag{3.22}$$

where e_k is statistically independent of W_k and satisfies $|e_k| < C(L, \mathcal{A}, n)$ with probability 1, where $0 < C(L, \mathcal{A}, n) < \infty$ is independent of α.

Proof. Let Z_{SIC} be the output of the algorithm, and let $\hat{Y} = Z_{\mathrm{SIC}}\mathcal{A}L$, and let $e_{\mathrm{SIC}} = Y - \hat{Y}$. As we have seen in Lemma 3.2, $e_{\mathrm{SIC}} \in \mathrm{CUBE}\cdot\mathcal{A}\cdot\operatorname{diag} L$. Inspecting the algorithm's successive rounding procedure, we have that

$$\begin{aligned}
Z_{\mathrm{SIC}} &= \operatorname{round}\left((Y - Z_{\mathrm{SIC}}\mathcal{A}(L - \operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.23)\\
&= \operatorname{round}\left((Y - Z_{\mathrm{SIC}}\mathcal{A}L\cdot L^{-1}(L - \operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.24)\\
&= \operatorname{round}\left((Y - \hat{Y}\cdot(I - L^{-1}\operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.25)\\
&= \operatorname{round}\left((Y - (Y - e_{\mathrm{SIC}})\cdot(I - L^{-1}\operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.26)\\
&= \operatorname{round}\left((YL^{-1}\operatorname{diag}(L) + e_{\mathrm{SIC}}\cdot(I - L^{-1}\operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.27)\\
&= \operatorname{round}\left((W\operatorname{diag}(L) + e_{\mathrm{SIC}}\cdot(I - L^{-1}\operatorname{diag}(L))) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right) && (3.28)\\
&= \operatorname{round}\left(W\mathcal{A}^{-1} + e_{\mathrm{SIC}}\cdot(I - L^{-1}\operatorname{diag}(L)) \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1}\right). && (3.29)
\end{aligned}$$

We are left with characterizing the random vector

$$e = e_{\mathrm{SIC}}\cdot\tilde{L} \ \cdot\ (\mathcal{A}\operatorname{diag}(L))^{-1} \tag{3.30}$$

where we defined the strictly lower triangular matrix $\tilde{L} = I - L^{-1}\operatorname{diag} L$. First, since $\tilde{L}$ is lower triangular, e_k is a deterministic function of $e_{\mathrm{SIC}}(k+1:n)$. Furthermore, by definition of the ZSIC algorithm $e_{\mathrm{SIC}}(k+1:n)$ is a deterministic function of $Y(k+1:n)$, and $Y(k+1:n)$ in turn, is a deterministic function of $W(k+1:n)$ since L is lower triangular. Since $W_k \perp\!\!\!\perp W(k+1:n)$, it therefore follows that $e_k \perp\!\!\!\perp W_k$ as claimed. It remains to upper bound $|e_k|$. To that end, let $S = e_{\mathrm{SIC}}\cdot(\mathcal{A}\operatorname{diag}(L))^{-1}$ and note that $S \in \mathrm{CUBE}$ since $e_{\mathrm{SIC}} \in \mathrm{CUBE}\cdot\mathcal{A}\cdot\operatorname{diag} L$. We therefore have that

$$e = S(\tilde{\mathcal{A}}\cdot\operatorname{diag} L)\tilde{L}(\tilde{\mathcal{A}}\operatorname{diag}(L))^{-1}. \tag{3.31}$$

is independent of α and has bounded support. □

Joint analysis of Huffman-GPTQ and PlainWaterSIC. Let us jointly analyze both the Huffman-GPTQ/GPTQ algorithm and the PlainWaterSIC algorithm. In both cases the spacing matrix will be of the form

$$\mathcal{A} = \alpha\tilde{\mathcal{A}}, \tag{3.32}$$

where $\tilde{\mathcal{A}} = I$ for GPTQ, whereas for PlainWaterSIC it will be

$$\tilde{\mathcal{A}} = \mathrm{diag}\left(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_n\right), \quad \text{where } \tilde{\alpha}_i = \frac{|L|^{1/n}}{|\ell_{ii}|} \; i = 1, \ldots, n. \tag{3.33}$$

In both cases $|\tilde{\mathcal{A}}| = 1$, and the density of the corresponding lattice is controlled only by the parameter α. We will show that

$$\lim_{\alpha \to 0} \frac{D(\alpha)}{\alpha^2} = \frac{1}{n} \frac{1}{12} \sum_{i=1}^{n} (\tilde{\alpha}_i \ell_{ii})^2. \tag{3.34}$$

and that

$$\lim_{\alpha \to 0} \left[H(Z_{\mathrm{SIC}}, i) - \frac{1}{2} \log(\alpha^2) \right] = \frac{1}{2} \log(2\pi e \sigma_W^2) - \log(\tilde{\alpha}_i), \quad i = 1, \ldots, n. \tag{3.35}$$

The expected rate $R(\alpha)$ is the result of applying entropy coding to each column $Z_{\mathrm{SIC},:,i}$, $i = 1, \ldots, n$ of Z_{SIC}. Since rows of W are independent, so are the a entries in each column $Z_{\mathrm{SIC},:,i}$ of Z_{SIC}. If a is sufficiently large, we can apply universal entropy coding on the ith column, $i = 1, \ldots, n$, and the expected number of bits will be $a(H(Z_{\mathrm{SIC},i}) + o(1))$, where $o(1)$ vanishes with a. We ignore this term in the remainder of the analysis since we take limit of $a \to \infty$. From (3.35) it therefore immediately follows that

$$\begin{aligned} \lim_{\alpha \to 0} [R(\alpha) - \frac{1}{2} \log(\alpha^2)] &= \frac{1}{n} \sum_{i=1}^{n} \lim_{\alpha \to 0} \left[H(Z_{\mathrm{SIC}}, i) - \frac{1}{2} \log(\alpha^2) \right] \\ &= \frac{1}{2} \log(2\pi e \sigma_W^2) - \frac{1}{n} \log |\tilde{\mathcal{A}}| \\ &= \frac{1}{2} \log(2\pi e \sigma_W^2), \end{aligned} \tag{3.36}$$

where the last equality follows from our assumption that $|\tilde{\mathcal{A}}| = 1$. Combining (3.34) and (3.36) we obtain

$$\lim_{\alpha \to 0} \left[R(\alpha) + \frac{1}{2} \log\left(D(\alpha)\right) \right] = \frac{1}{2} \log \left(\frac{2\pi e}{12} \frac{1}{n} \sum_{i=1}^{n} (\tilde{\alpha}_i \ell_{ii})^2 \sigma_W^2 \right). \tag{3.37}$$

Subtracting $R_{\mathrm{WF}}(D(\alpha), \Sigma_X) = R_{\mathrm{High-Rate}}(D(\alpha), \Sigma)$ (since $D(\alpha) \to 0$ as $\alpha \to 0$) from both sides, and recalling that

$$|\Sigma_X|^{1/n} = |L|^{2/n} = |L\tilde{\mathcal{A}}|^{2/n} = \prod_{i=1}^{n} (\tilde{\alpha}_i \ell_{ii})^{2/n},$$

since $|\tilde{\mathcal{A}}| = 1$, we obtain

$$\lim_{\alpha \to 0} [R(\alpha) - R_{\mathrm{WF}}(D(\alpha), \Sigma_X)] = \frac{1}{2} \log \left(\frac{2\pi e}{12} \frac{\frac{1}{n} \sum_{i=1}^{n} (\tilde{\alpha}_i \ell_{ii})^2}{\prod_{i=1}^{n} (\tilde{\alpha}_i \ell_{ii})^{2/n}} \right). \tag{3.38}$$

The claimed result now follows from substituting the corresponding $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_n$ for GPTQ and for WaterSIC. It therefore only remains to prove (3.34) and (3.35).

Proof of (3.34). Consider the lattice $\Lambda = \alpha\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}$. Recall that $e_{\rm SIC} = WL - Z_{\rm SIC}\alpha\tilde{\mathcal{A}}L$ and that $e_{\rm SIC} \in \alpha\mathrm{CUBE}\cdot\tilde{\mathcal{A}}\,\mathrm{diag}(L)$. Therefore,

$$e_{\rm SIC} = e_W L \tag{3.39}$$

where

$$e_W = W - \alpha Z_{\rm SIC}\tilde{\mathcal{A}}, \tag{3.40}$$

and $Z_{\rm SIC} \in \mathbb{Z}^{1\times n}$ is chosen such that $e_W \in \alpha\mathrm{CUBE}\cdot\tilde{\mathcal{A}}\,\mathrm{diag}(L)L^{-1} \triangleq \mathcal{P}$. We will show that e_W is nearly uniform on $\mathcal{P}$, and it will then follow that so is $e_{\rm SIC}$.

Let $f_{e_W}(x)$ denote the probability density function (pdf) of the random vector e_W. Note that

$$f_{e_W}(x) = f_{\sigma_W,\tilde{\Lambda}}(x)\mathbb{1}\{x\in\mathcal{P}\}, \tag{3.41}$$

where $\tilde{\Lambda} = \alpha\mathbb{Z}^{1\times n}\tilde{\mathcal{A}}$. By definition of the flatness factor we have that

$$f_{e_W}(x) \in \left[1 \pm \epsilon_{\tilde{\Lambda}}(\sigma_W)\right]\frac{1}{\mathrm{Vol}(\mathcal{P})}\mathbb{1}\{x\in\mathcal{P}\} \tag{3.42}$$

It therefore follows that the pdf of $e_{\rm SIC} = e_W L$ satisfies

$$f_{e_{\rm SIC}}(x) \in \left[1 \pm \epsilon_{\tilde{\Lambda}}(\sigma_W)\right]\frac{1}{\mathrm{Vol}(\mathcal{P}L)}\mathbb{1}\{x\in\mathcal{P}L\}. \tag{3.43}$$

Consequently,

$$\begin{aligned} D = \frac{1}{n}\mathbb{E}\|e_{\rm SIC}\|^2 &= \int_{x\in\mathcal{P}L}\|x\|^2 f_{e_{\rm SIC}}(x)dx \\ &\in \left[1 \pm \epsilon_{\tilde{\Lambda}}(\sigma_W)\right]\mathbb{E}\|U_{\mathcal{P}L}\|^2 \qquad (3.44)\\ &\in \left[1 \pm \epsilon_{\tilde{\Lambda}}(\sigma_W)\right]\frac{1}{n}\frac{\alpha^2}{12}\sum_{i=1}^{n}(\tilde{\alpha}_i\ell_{ii})^2, \qquad (3.45) \end{aligned}$$

where $U_{\mathcal{P}L} \sim \mathrm{Uniform}(\mathcal{P}L)$. By (3.21) we have that $\lim_{\alpha\to 0}\epsilon_{\tilde{\Lambda}}(\sigma_W) = 0$ and (3.34) indeed holds.

Proof of (3.35). By Lemma 3.5 we have

$$Z_{{\rm SIC},i} = \mathrm{round}\left(\frac{1}{\alpha}V_\alpha\right), \tag{3.46}$$

where

$$V_\alpha = \frac{W_i}{\tilde{\alpha}_k} + \alpha e_i. \tag{3.47}$$

Here $W_i \sim \mathcal{N}(0, \sigma_W^2)$, and e_i is bounded in an interval independent of α with probability 1, and is statistically independent of W_i. For $\Delta > 0$ and a random variable X let $X^\Delta = \Delta \cdot \mathrm{round}\left(\frac{X}{\Delta}\right)$. A classic result of Rényi [42, Theorem 1], [24, eq. 2.21] shows that if $h(X)$ exists and $H(\mathrm{round}(X)) < \infty$ then

$$\lim_{M\to\infty} H(X^{1/M}) - \log M = h(X) + E(M, P_X), \tag{3.48}$$

where $E(M, P_X)$ vanishes with M. Let $\mathcal{V}_{\alpha_0}$ be the family of distributions on V_α for all $\alpha < \alpha_0$. Inspecting the proof of [42, Theorem 1], we see that for sufficiently small α_0 his equations (40-41) can be made to hold simultaneously with the same $L(\varepsilon)$ for all distributions in $\mathcal{V}_{\alpha_0}$. It therefore follows that $E(M, P_X)$ converges to 0 with $M \to \infty$ uniformly for all $P_X \in \mathcal{V}_{\alpha_0}$. Since $H(\mathrm{round}(\frac{1}{\alpha} \cdot V_\alpha) = H\left(V_\alpha^\alpha\right)$ we have

$$\lim_{M\to\infty} \left[H(Z_{\mathrm{SIC},i}) - \log(\alpha)\right]\Big|_{\alpha=1/M} = \lim_{M\to\infty} h(V_{1/M}). \tag{3.49}$$

Finally, using the continuity of entropy, specifically [25] that for $e \perp\!\!\!\perp X$ where $\mathbb{E}(e^2) < \infty$ and $h(X)$ exists it holds that

$$\lim_{\alpha\to 0}[h(X + \alpha e)] = h(X), \tag{3.50}$$

we obtain

$$\lim_{\alpha\to 0} \left[H(Z_{\mathrm{SIC},i}) - \log(\alpha)\right] = \frac{1}{2}\log(2\pi e \sigma_W^2) - \log(\tilde{\alpha}_i), \tag{3.51}$$

as claimed.

This completes the proof of Theorem 3.3. To summarize the chapter: any weight-only quantizer must respect the waterfilling lower bound $R_{\mathrm{WF}}(D, \Sigma_X)$ (Proposition 3.1); Huffman-GPTQ leaves a rate gap that grows as $\frac{1}{2}\log(\frac{1}{n}\sum_i \ell_{ii}^2 / \prod_i \ell_{ii}^{2/n})$ and is unbounded in the worst case; whereas PlainWaterSIC, by allocating per-column grid spacings $\alpha_i \propto 1/\ell_{ii}$, attains the IT lower bound up to a uniform gap of $\frac{1}{2}\log_2(2\pi e/12) \approx 0.255$ bits, the standard high-rate price of scalar uniform quantization. Deploying this rate allocation on real LLM weight matrices is the subject of the next chapter.

Chapter 4

The WaterSIC Algorithm

The previous chapter introduced PlainWaterSIC, a conceptual algorithm that achieves the rate-distortion guarantee of Theorem 3.3 on iid Gaussian weights and exact statistics. PlainWaterSIC is not yet a deployable LLM quantizer; the present chapter describes the modifications needed to obtain a practical algorithm that performs well on real weight matrices. As in Chapter 3, we factor $\Sigma_X = LL^\top$ with L lower-triangular, and given a constant c we set the i-th column grid spacing to $\alpha_i = c/\ell_{ii}$. We then apply a sequence of corrections, each addressing a specific way in which the assumptions of PlainWaterSIC fail on real LLMs: per-column refinements (Section 4.1), joint quantization of attention projections (Section 4.2), adaptive mixing of unquantized and drift-corrected covariances (Section 4.3), entropy coding and rate control (Section 4.4), and the WaterSIC-FT finetuning variant (Section 4.5). The full algorithm is given as Algorithm 3, with the rescaler optimization spelled out separately as Algorithm 4; an ablation isolating each component is deferred to Appendix B.

4.1 Per-Column Refinements

We now describe four refinements applied to each column of the weight matrix during quantization. The first three (LMMSE correction, activation drift correction, residual stream correction) modify the input seen by the quantizer to reduce layer-output distortion; the fourth (diagonal rescalers) post-processes the integer codes. Specific values of all hyperparameters introduced below, including the per-layer mixing coefficients of Section 4.3, are reported in Appendix A.

LMMSE correction. When ZSIC (Algorithm 1) calls a round$(\cdot)$ on a i-th column of the matrix $Y = WL$, it effectively solves

$$\operatorname*{argmin}_{z\in\mathbb{Z}^n} \sum_{k=1}^{a} (Y_{k,i} - z_k L_{i,i}\alpha_i)^2 = \text{round}(Y_{:,i}/c)$$

and results in the approximation $Y_{:,i} \approx zc$. However, since entries of $Y_{:,i}$ are typically unimodal with a mode at 0, the rounding errors tend to be biased away from 0. Thus, a better reconstruction of $Y_{:,i}/c$ can be obtained by applying an extra shrinkage factor γ_i replacing zc with $\gamma_i zc$, where scalar γ_i (known as linear-MMSE, or LMMSE correction) can be chosen

via solving a simple quadratic equation:

$$\gamma_i = \operatorname*{argmin}_{\gamma \in \mathbb{R}} \sum_{k=1}^{a} (Y_{k,i} - \gamma c z_k)^2 = \frac{\sum_k Y_{k,i} z_k}{c \sum_k z_k^2} \,. \tag{4.1}$$

Importantly, recursive ZSIC adjustment to Y should use the γ_i-corrected value too: $Y \leftarrow Y - \gamma_i c L_{i,:} z$.

Note that overall we see that the final weight $\widehat{w}_i \in (\alpha_i \gamma_i)\mathbb{Z}^n$ and since $\gamma_i < 1$ one may wonder why not use the finer grid $(\alpha_i \gamma_i)$ from the start? The answer is that reducing the grid spacing will increase the entropy of $\widehat{w}_i$ (and rate). This is one of the counter-intuitive effects in the low-rate regime: it is unnatural, but important, to dequantize to elements that are distinct from the quantization ones (unless one uses a sophisticated vector quantizer, i.e. a centroidal Voronoi tessellation (CVT), such as returned by a fully converged Lloyd-Max algorithm).

Activation drift correction (Qronos/QA-LDLQ). Several works [23] (Section 4.5), [30], [31] simultaneously noticed the following discrepancy in the original formulation: instead of minimizing $\mathbb{E}[\|WX - \hat{W}X\|_2^2]$ we should be minimizing

$$\min_{\hat{W}} \mathbb{E}[\|WX - \hat{W}\hat{X}\|_2^2] \,,$$

where $\hat{X}$ is the input activation in the quantized model (i.e. due to quantization of previous layers, the input $\hat{X}$ to the present layer is different from X in the unquantized model). This problem after some algebra reduces to

$$\min_{\hat{W}} \|\hat{y} - \hat{W}\hat{L}\|_2^2 \,, \tag{4.2}$$

where $\Sigma_{\hat{X}} = \mathbb{E}[\hat{X}\hat{X}^\top]$ and its Cholesky decomposition is $\Sigma_{\hat{X}} = \hat{L}\hat{L}^\top$, $\Sigma_{X,\hat{X}} = \mathbb{E}[X\hat{X}^\top]$ and

$$\hat{y} = W\Sigma_{X\hat{X}}(\hat{L}^\top)^{-1} \,. \tag{4.3}$$

Thus, the only modification to the previous discussion is the replacement of the Hessian and adjusted value of y.

Residual stream correction. We noticed that some of the hardest to compress layers are the downprojection layers (wo for attention and w2 for FFN). Part of the reason is that they in fact do not produce output $Y = WX$ as we have been assuming, but rather $Y = WX + R$, where R is the state of the residual stream to which the downprojection contributes to. For this reason, we should really be solving the problem

$$\min_{\hat{W}} \mathbb{E}[\|WX + R - (\hat{W}\hat{X} + \hat{R})\|_2^2] \,,$$

where $\hat{X}, \hat{R}$ are the input activations and the state of residual stream in the quantized model. In this case, after introducing the $\Sigma_{\Delta,\hat{X}} = \mathbb{E}[(R - \hat{R})\hat{X}^\top]$ we can see that the problem reduces to (4.2) but with

$$\hat{y} = (W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})(\hat{L}^\top)^{-1} \,. \tag{4.4}$$

Diagonal rescalers. The final optimization we would like to do is another round of row/column rescalers. Specifically, after running the ZSIC algorithm, we obtain a diagonal matrix $\mathcal{A}$ and an integer matrix Z_{SIC}. We set $\hat{W}_0 = Z_{\text{SIC}}\mathcal{A}$ as our preliminary estimate and search the final optimizer in the form

$$\hat{W} = T\hat{W}_0\Gamma\,,$$

where $T = \operatorname{diag}(t_1, \dots, t_a)$ and $\Gamma = \operatorname{diag}(\gamma_1, \dots, \gamma_n)$. Because of scale invariance, we will also normalize the matrices such that $\operatorname{tr} T = a$. Initially, we set $T = I$ and Γ is taken from values γ_i obtained in (4.1).

The loss objective to be minimized in terms of T and Γ is

$$\ell(T, \Gamma) = \operatorname{tr}(W\Sigma_X W^\top - 2(W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})\Gamma\hat{W}_0^\top T)\,.$$

For a fixed Γ the loss in terms of $T = \operatorname{diag}(t_1, \dots, t_a)$ is given by

$$\sum_{i=1}^{a} t_i^2 F_{i,i}^{(8)} - 2F_{i,i}^{(7t)} t_i\,,$$

where $F^{(8)} = \hat{W}_0\Gamma\Sigma_{\hat{X}}\Gamma\hat{W}_0^\top$, $F^{(7t)} = (W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})\Gamma\hat{W}_0^\top$. Thus, the optimal value of t_i can be given by $t_i = F_{i,i}^{(7t)}/F_{i,i}^{(8)}$.

On the other hand, for a fixed T the loss function in terms of $\Gamma = \operatorname{diag}(\gamma_1, \dots, \gamma_n)$ is given by

$$\sum_{i,j} \gamma_i\gamma_j F_{i,j}^{(3)} - 2\sum_i F_{i,i}^{(4)}\gamma_i\,,$$

where $F^{(3)} = F^{(2)} \odot \Sigma_{\hat{X}}$, $F^{(2)} = \hat{W}_0^\top T^2 \hat{W}_0$, $F^{(4)} = \hat{W}_0^\top T(W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})$. Note that by Schur's product theorem $F^{(3)}$ is positive definite whenever $F^{(2)}$ and $\Sigma_{\hat{X}}$ are. Thus, minimizer of this expression exists and is given by

$$(\gamma_1, \dots, \gamma_n)^\top = F^{(3)\,-1}\operatorname{diag}(F^{(4)})\,.$$

Thus by alternating T and Γ steps, we can progressively improve the loss attained by the $\hat{W}$. (We also need to renormalize T and Γ after each step so that $\operatorname{tr} T = a$ is satisfied.) Note also that adding a row rescaler (in BF16) adds $16/a$ overhead to the final rate (and recall that $16/n$ is paid for communicating $\mathcal{A}$, which we fuse into $\mathcal{A}\Gamma$).

See Fig. B.1 for representation of typical values of T and Γ.

4.2 Joint Quantization of Attention Projections

The refinements of Section 4.1 treat each linear layer in isolation, but the attention projections w_q, w_k, w_v interact through the softmax in a way that the layerwise MSE objective fails to capture. This section describes two attention-specific adjustments: a calibration covariance weighted by per-token attention importance, and an adaptive mixing scheme that interpolates between the weighted, drift-corrected covariances and the unweighted, unquantized ones, with mixing coefficients selected by golden-section search (Section 4.3).

Attention-weighted calibration. The covariance matrices Σ_X, $\Sigma_{\hat{X}}$, and $\Sigma_{X,\hat{X}}$ above are estimated by averaging uniformly over all token positions. When quantizing attention matrices W_Q, W_K, W_V, however, we found that some tokens need to be quantized with higher fidelity. To correct for this, we weight the covariance estimates for QKV projections by a per-token attention importance score:

$$p_j := \frac{1}{N_H\,(T-j)} \sum_{h=1}^{N_H} \sum_{i=j}^{T-1} \alpha_{h,i,j}\,. \tag{4.5}$$

Here, $\alpha_{h,i,j}$ is the attention probability from query i to key j in head h, computed from the unquantized model. The weighted covariances $\Sigma_X^{(w)} = \mathbb{E}\Big[\sum_j p_j\, x_j x_j^\top\Big]$ (and analogously $\Sigma_{\hat{X}}^{(w)}$ and $\Sigma_{X,\hat{X}}^{(w)}$) are then substituted into (4.2)-(4.3). This weighting applies only to QKV projections.

Adaptive mixing. We noticed that when accumulated prior layers introduce anomalously large discrepancy between quantized $\hat{X}$ and unquantized X inputs, the drift compensation and attention weighting become overly fixated on correcting this discrepancy, thus losing fidelity on truly relevant features of X. To fix this we can replace $\hat{X}$ with X with a certain probability ϵ_{qr} during calibration calculation. Similarly, we introduce parameter ϵ_{aw} that determines the amount of attention-weighting. Overall, the actual calibration matrices we use are given by

$$\begin{aligned}
\Sigma_X^{(\text{final})} &:= (1-\epsilon_{\text{aw}})\,\Sigma_X^{(w)} + \epsilon_{\text{aw}}\Sigma_X\,,\\
\Sigma_\bullet^{(\text{final})} &:= (1-\epsilon_{\text{aw}})\,((1-\epsilon_{\text{qr}})\Sigma_\bullet^{(w)} + \epsilon_{\text{qr}}\Sigma_X^{(w)}) +\\
&\qquad \epsilon_{\text{aw}}((1-\epsilon_{\text{qr}})\Sigma_\bullet + \epsilon_{\text{qr}}\Sigma_X)\,,
\end{aligned} \tag{4.6}$$

where $\bullet$ in the last equation stands for $\hat{X}$ or $X, \hat{X}$ (that is, gives equation for effective $\Sigma_{\hat{X}}$ and $\Sigma_{X,\hat{X}}$). Parameters ϵ_{qr} and ϵ_{aw} are not set *a priori* but are optimized by golden-section search for each layer, see details in Section 4.3 and Tables A.1, A.2.

We only applied mixing optimizations for attention projections (w_q, w_k, w_v), for the rest $\epsilon_{\text{qr}} = 0$ and $\epsilon_{\text{aw}} = 1$.

4.3 Calibration Statistics and Adaptive Mixing

This section describes how the covariance estimates of the previous two sections are collected from calibration data, how the per-layer mixing coefficients $(\epsilon_{\text{qr}}, \epsilon_{\text{aw}})$ are selected, and how Hessian damping interacts with the resulting blended matrices.

Collecting raw calibration data. For calibration and tuning, we use the full WikiText-2 training split. We concatenate the text into a single token stream (with a single BOS token prepended) and partition it into non-overlapping sequences of length 2048, discarding any remainder. The exact number of sequences depends on the tokenizer (e.g., ≈ 1189 for Llama-3.2-1B). Only the first sequence begins with a BOS token; subsequent sequences start at arbitrary token boundaries. We verified on Llama-3.2-1B that prepending a BOS token to *every* sequence affects neither perplexity nor the resulting calibration statistics.

Given these sequences, and assuming that preceding layers are already quantized, we run both the unquantized and (partially) quantized models to collect the layer inputs X and $\hat{X}$, as well as the residual stream states R and $\hat{R}$. We then form the required (cross-)covariance estimates by averaging over all token positions (and their attention-weighted counterparts, where applicable).

Selecting adaptive mixing coefficients. As described in (4.6) for some layers we undertake additional optimization in the form of mixing unquantized statistics into quantized one.

Specifically, first we introduce a mixing parameter $\epsilon_{\text{qr}} \in [0, 1]$ that interpolates between the drift-corrected and original statistics:

$$\begin{aligned} \Sigma_{\hat{X}}^{(\text{mix})} &= (1 - \epsilon_{\text{qr}})\, \Sigma_{\hat{X}} + \epsilon_{\text{qr}}\, \Sigma_X, \\ \Sigma_{X,\hat{X}}^{(\text{mix})} &= (1 - \epsilon_{\text{qr}})\, \Sigma_{X,\hat{X}} + \epsilon_{\text{qr}}\, \Sigma_X. \end{aligned} \tag{4.7}$$

Setting $\epsilon_{\text{qr}} = 0$ recovers full drift correction (4.2), while $\epsilon_{\text{qr}} = 1$ falls back to the original (unquantized) Hessian.

Once the drift mixing is fixed, we apply attention weighting on top of the resulting covariances. Let $\Sigma_\bullet^{(w)}$ denote the attention-weighted estimate from (4.5) computed from the mixed statistics $\Sigma_\bullet^{(\text{mix})}$. A second parameter $\epsilon_{\text{aw}} \in [0, 1]$ interpolates between the weighted and uniform versions:

$$\begin{aligned} \Sigma_\bullet^{(\text{final})} := (1 - \epsilon_{\text{aw}})\, \Sigma_\bullet^{(w)} + \epsilon_{\text{aw}}\, \Sigma_\bullet^{(\text{mix})}, \\ \Sigma_\bullet \in \{\Sigma_X,\, \Sigma_{\hat{X}},\, \Sigma_{X,\hat{X}}\}. \end{aligned} \tag{4.8}$$

Thus, the attention importance scores weigh the already drift-mixed covariances, and ϵ_{aw} controls how strongly this reweighting is applied.

We optimize $(\epsilon_{\text{qr}}, \epsilon_{\text{aw}})$ per layer via a lightweight coordinate search. For a given layer, let $\hat{w}_q, \hat{w}_k, \hat{w}_v$ denote the quantized projections obtained using the blended statistics (4.7)-(4.8). We minimize the relative MSE of the input to w_o, i.e., the output of the multi-head attention block before the output projection:

$$\min_{\epsilon_{\text{qr}},\, \epsilon_{\text{aw}} \in [0,1]} \frac{\mathbb{E}\left[\left\|\text{Attn}(X;\, w_q, w_k, w_v) - \text{Attn}(\hat{X};\, \hat{w}_q, \hat{w}_k, \hat{w}_v)\right\|_F^2\right]}{\mathbb{E}\left[\left\|\text{Attn}(X;\, w_q, w_k, w_v)\right\|_F^2\right]}, \tag{4.9}$$

where $\text{Attn}(X;\, w_q, w_k, w_v)$ denotes the multi-head self-attention output (including the softmax) given input activations X and projection weights, and $\hat{X}$ are the (already quantized) activations produced by the preceding layers. Measuring distortion at the w_o input rather than at individual projection outputs captures the error amplification through the softmax nonlinearity, which empirically dominates the quantization loss in attention layers.

The objective in (4.9) is empirically unimodal in each coordinate when the other is held fixed. We therefore first optimize ϵ_{qr} with ϵ_{aw} at a default value using golden-section search over $[0, 1]$, then optimize ϵ_{aw} with ϵ_{qr} fixed at its optimum. Each evaluation requantizes (w_q, w_k, w_v) jointly and performs a forward pass through the attention block on

the calibration set. The resulting parameters $(\epsilon^\star_{\rm qr}, \epsilon^\star_{\rm aw})$ are stored per layer and used during the final quantization pass.

Hessian damping. It is a common practice starting from GPTQ codebase to replace

$$\Sigma_X \leftarrow \Sigma_X + \delta I_n \,,$$

where δ is a hyperparameter determining regularization strength of the collected covariance matrix (by default, $\delta = \frac{0.1}{n} \operatorname{tr} \Sigma_X$). Note that adding this term is equivalent to replacing loss objective by

$$\min_{\hat{W}} \operatorname{tr}(W - \hat{W})\Sigma_X(W - \hat{W})^\top + \delta \|W - \hat{W}\|_F^2 ,$$

which evidently safeguards $\hat{W}$ from deviating from W too much.

Thus, in the presence of drift correction and residual compensation we are to minimize objective:

$$\min_{\hat{W}} \mathbb{E}[\|WX + R - (\hat{W}\hat{X} + \hat{R})\|_2^2] + \delta \|W - \hat{W}\|_F^2 ,$$

which is equivalent to modifying

$$\begin{aligned} \Sigma_X &\leftarrow \Sigma_X + \delta I_n \\ \Sigma_{\hat{X}} &\leftarrow \Sigma_{\hat{X}} + \delta I_n \\ \Sigma_{X,\hat{X}} &\leftarrow \Sigma_{X,\hat{X}} + \delta I_n \qquad \text{(note!)} \\ \Sigma_{\Delta,\hat{X}} &\leftarrow \Sigma_{\Delta,\hat{X}} \qquad \text{(not a typo!)} \end{aligned}$$

Damping δI is applied after the adaptive mixing, on the resulting blended matrices.

4.4 Implementation Details

We first describe how the integer codes produced by Algorithm 1 are entropy-coded and how the rate is controlled in practice, then turn to two numerical considerations that arise on real LLMs.

Entropy coding. In Algorithm 2 we performed entropy coding on each column of $Z_{\rm SIC}$ separately. This makes sense because for iid rows of W, each column of $Z_{\rm SIC}$ is iid, but the distribution may differ from column to column. In practice, however, we observed that performing entropy coding jointly on the entire matrix $Z_{\rm SIC}$ produces a negligible increase in rate. We therefore adopt the latter option in the practical WaterSIC algorithm. Note that this does not imply that most columns are coded to the same rate, which is indeed not the case (see Fig. B.2).

We also checked that instead of entropy coding, one might use LZMA or Zstd compressor and indeed obtain the same number of bits as predicted by our estimate of entropy over entries. We report these findings in Appendix B.

Rate assignment. To achieve a desired target rate for the layer, we note that the final entropy is a monotone function of the constant c, which is approximately linear with a slope close to unity. We exploit this structure via a secant method: evaluating at the target rate reveals an offset, which a secant correction refines in about 3 iterations to within < 0.005 bits

of the target. For computational efficiency, the entropy estimate uses a randomly sampled fraction of the rows. We also maintain a global budget that can be allocated to the remaining unquantized layers, which mitigates discrepancies between the estimated and actual entropy.

Dead feature erasure. We found that occasionally some input dimensions of X have near-zero variance, i.e., $[\Sigma_X]_{ii} \approx 0$. These "dead features" cause numerical issues in the Cholesky decomposition of $\Sigma_{\hat{X}}$ and can lead to unstable rescaler optimization, while carrying negligible signal. Traditionally, these were handled by *damping*, which adds a multiple of identity to certain matrices (see Section 4.3), but we apply a more direct workaround. We declare dimension i dead if $[\Sigma_X]_{ii} < \tau\tilde{\sigma}^2$, where $\tilde{\sigma}^2 := \text{median}_j\,[\Sigma_X]_{jj}$ and $\tau = 10^{-3}$. We use the median rather than the mean because a few high-variance dimensions (e.g., in SiLU-gated intermediate activations) can inflate the mean by orders of magnitude, causing the threshold to erroneously flag the majority of dimensions as dead. We then set the corresponding columns of W to zero and apply our quantization pipeline to the reduced system obtained by removing these dimensions. The quantized weight is expanded back to the original size by inserting zero-columns at the dead features. In early layers, the number of dead dimensions can be substantial (e.g., due to layer normalization suppressing certain coordinates), which both improves the numerical stability of the Cholesky factor $\hat{L}$ and frees quantization rate budget for the remaining live dimensions.

4.5 Post-Quantization Finetuning

After the quantization pipeline of the previous sections produces integer codes Z and initial rescalers T, Γ, we can further improve quality by finetuning the continuous parameters $t_1, \ldots, t_a$ and $\gamma_1, \ldots, \gamma_n$ via gradient descent, while keeping the integer codes Z frozen. Since the dequantized weight $\hat{W} = T(Z \odot \alpha)\Gamma$ is fully differentiable with respect to t and γ, no straight-through estimator is needed. We minimize the KL divergence between the unquantized (teacher) and quantized (student) model outputs:

$$\mathcal{L} = \text{KL}(p_{\text{teacher}}(\cdot \mid x) \,\|\, p_{\text{student}}(\cdot \mid x))$$

using AdamW with cosine annealing. The total number of trainable parameters is $a + n$ per layer (two vectors), which is negligible compared to the an frozen integer codes.

In the plots and tables that follow, we distinguish between WaterSIC and WaterSIC-FT, where the latter applies this post-quantization finetuning and the former does not.

The full algorithm, combining the elements of this chapter, is given as Algorithm 3, with the rescaler optimization spelled out in Algorithm 4. Appendix B analyzes how the individual techniques contribute to overall quantization quality, reporting relative MSE plots for the inputs to each layer's weight matrices. With the algorithm fully specified, the next chapter evaluates WaterSIC across the Llama and Qwen families and shows that the theoretical gains predicted by Chapter 3 translate into a new rate-distortion frontier on real LLMs.

Algorithm 3 WaterSIC: full algorithm details

Require: Weight matrix $W \in \mathbb{R}^{a \times n}$, covariance $\Sigma_X \in \mathbb{R}^{n \times n}$, scale parameter $c > 0$, damping $\delta \geq 0$.

Require: Covariance $\Sigma_{\hat{X}}$, $\Sigma_{X,\hat{X}}$, $\Sigma_{\Delta,\hat{X}}$ $\triangleright$ If not available, set first two to Σ_X and the last one to 0.

Ensure: Reconstructed weights $\hat{W}$, effective rate R_{eff}

Phase 1: Setup

$H \leftarrow \Sigma_{\hat{X}} + \delta \cdot \text{mean}(\text{diag}(\Sigma_{\hat{X}})) \cdot I_n$ $\triangleright$ For stability in early layers

$L \leftarrow \text{chol}(H)$ $\triangleright$ Lower-triangular: $H = LL^\top$

$Y \leftarrow (W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})\,(L^\top)^{-1}$ $\triangleright$ Use triangular solver for efficiency

$\alpha_k \leftarrow c/L_{k,k}$ for $k = 1, \ldots, n$

Phase 2: ZSIC with LMMSE correction

for $i = n, n{-}1, \ldots, 1$ **do**

  $\mathbf{z}_i \leftarrow \text{round}(Y_{:,i} \,/\, c)$

  $\gamma_i \leftarrow \mathbf{z}_i^\top Y_{:,i} / (c\|\mathbf{z}_k\|^2)$

  $Y \leftarrow Y - \gamma_k \alpha_k \, \mathbf{z}_k \, \mathbf{l}_k^\top$ $\triangleright$ $\mathbf{l}_k^\top$: row k of L

end for

Phase 3: Rate computation

$\mathcal{H} \leftarrow -\sum_v p_v \log_2 p_v, \quad p_v = |\{(i,k) : Z_{ik} = v\}|/(an)$

$R_{\text{eff}} \leftarrow \mathcal{H} + 16/a + 16/n$ $\triangleright$ Entropy + side-info overhead

Phase 4: Diagonal rescaler optimization

$\hat{W}_0 \leftarrow Z\,\text{diag}(\alpha)$

$T, \Gamma \leftarrow$ FindOptimalRescalers($\hat{W}_0,\ W,\ \Sigma_X,\ \Sigma_{\hat{X}},\ \Sigma_{X,\hat{X}},\ \Sigma_{\Delta,\hat{X}};\ \gamma_{\text{init}} = \gamma$)

$\hat{W} \leftarrow TZ\Gamma\ \text{diag}(\alpha)$ $\triangleright$ $T = \text{diag}(t),\ \Gamma = \text{diag}(\tilde{\gamma})$

return $R_{\text{eff}},\ \hat{W}$

Algorithm 4 FindOptimalRescalers: Alternating optimization of diagonal row and column rescalers

Require: $\hat{W}_0 \in \mathbb{R}^{a\times n}$ (pre-rescaler reconstruction), $W \in \mathbb{R}^{a\times n}$ (original weights), $\Sigma_X, \Sigma_{\hat{X}}, \Sigma_{X,\hat{X}}, \Sigma_{\Delta,\hat{X}} \in \mathbb{R}^{n\times n}$, initial $\gamma^{(0)} \in \mathbb{R}^n$, tolerance ε, ridge $\lambda \geq 0$
Ensure: Diagonal matrices $T = \mathrm{diag}(t)$, $\Gamma = \mathrm{diag}(\gamma)$

// **Objective:** $\mathcal{J}(T,\Gamma) = \frac{1}{an}\,\mathrm{tr}\big(W\Sigma_X W^\top - 2\,(W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})(T\hat{W}_0\Gamma)^\top + T\hat{W}_0\Gamma\,\Sigma_{\hat{X}}\,\Gamma\hat{W}_0^\top T\big)$

$t \leftarrow \mathbf{1}_a$; $\gamma \leftarrow \gamma^{(0)}$
$s \leftarrow \|t\|_1/a$; $t \leftarrow t/s$; $\gamma \leftarrow s\,\gamma$ ▷ Normalize: $\|t\|_1 = a$
$\mathcal{L}_{\mathrm{prev}} \leftarrow \mathcal{J}(\mathrm{diag}(t), \mathrm{diag}(\gamma))$
for iter $= 1, 2, \ldots$ **do**
Γ-step:
$F \leftarrow \hat{W}_0^\top \mathrm{diag}(t^2)\hat{W}_0$ ▷ $n \times n$
$G \leftarrow \Sigma_{\hat{X}} \odot F$ ▷ Hadamard product, $n \times n$
$\mathbf{d} \leftarrow \mathrm{diag}\big(\hat{W}_0^\top \mathrm{diag}(t)\,(W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})\big)$ ▷ n-vector
$\gamma \leftarrow (G + \lambda I_n)^{-1}\mathbf{d}$
T-step:
$\mathbf{p} \leftarrow \mathrm{diag}\big((W\Sigma_{X,\hat{X}} + \Sigma_{\Delta,\hat{X}})\ \mathrm{diag}(\gamma)\,\hat{W}_0^\top\big)$ ▷ a-vector
$\mathbf{q} \leftarrow \mathrm{diag}\big(\hat{W}_0\ \mathrm{diag}(\gamma)\,\Sigma_{\hat{X}}\ \mathrm{diag}(\gamma)\,\hat{W}_0^\top\big)$ ▷ a-vector
$t_i \leftarrow p_i\,/\,(q_i + \lambda)$ for $i = 1, \ldots, a$
Re-normalize and check convergence:
$s \leftarrow \|t\|_1/a$; $t \leftarrow t/s$; $\gamma \leftarrow s\,\gamma$
$\mathcal{L}_{\mathrm{curr}} \leftarrow \mathcal{J}(\mathrm{diag}(t), \mathrm{diag}(\gamma))$
if $|\mathcal{L}_{\mathrm{curr}} - \mathcal{L}_{\mathrm{prev}}|\,/\,(|\mathcal{L}_{\mathrm{prev}}| + 10^{-12}) < \varepsilon$ **then**
break
end if
$\mathcal{L}_{\mathrm{prev}} \leftarrow \mathcal{L}_{\mathrm{curr}}$
end for
return $T = \mathrm{diag}(t)$, $\Gamma = \mathrm{diag}(\gamma)$

Chapter 5

Evaluation Results and Limitations

We evaluate WaterSIC on the Llama and Qwen families across model sizes from 1B to 70B parameters and rates from 1 to 4 bits per weight. Section 5.1 reports rate-distortion curves per model; Section 5.2 isolates the effect of the calibration and finetuning corpus on Llama-3.2-1B at 2 bits; Section 5.3 discusses limitations and future work.

5.1 Per-Model Results

We evaluate WaterSIC on five base models spanning three families and three orders of magnitude in parameter count: Llama-3.2-1B, Qwen3-8B, Llama-3-8B, Llama-2-7B, and Llama-3-70B. The first four are evaluated at every bitrate from 1 to 4 bits per weight; the largest (Llama-3-70B) is reported only at 4 bits, as a preliminary result. At every (model, bitrate) configuration tested, WaterSIC matches or exceeds the strongest published baseline at comparable rate, and the gains are largest in the aggressive low-bit regime (1 to 2.5 bits) where the IT-optimal rate allocation matters most. Figure 1.1 previews the headline rate-distortion comparison across all models.

Throughout this section we follow a uniform protocol: WaterSIC is calibrated on the WikiText-2 training split at context length 2048, and WaterSIC-FT denotes the same model after post-quantization finetuning on WikiText-2 at the same context length. We evaluate on both WikiText-2 and C4 test perplexity. Because both calibration and finetuning use WikiText-2, the C4 gap to the BF16 reference is wider than the W2 gap at low rates, most visibly at 1 and 1.5 bits. This is a domain-matching effect, not a property of WaterSIC; Section 5.2 isolates it by varying the corpora across WikiText-2, RedPajama, and C4 on Llama-3.2-1B at 2 bits.

On rate conventions: WaterSIC and other entropy-coded baselines (Huffman-GPTQ, Huffman-RTN) report rates in bits per weight via entropy of the integer codes, while several prior baselines report rates via log-cardinality of their codebook. We follow whichever convention each reference uses and label axes accordingly.

5.1.1 Llama-3.2-1B

Figure 5.1 and Table 5.1 show the perplexity-rate tradeoff for Llama-3.2-1B. We compare WaterSIC to a diverse set of PTQ baselines: AWQ [12], Huffman-GPTQ [1], NestQuant [23], and QTIP [9]. Across the range of rates we consider, WaterSIC improves upon the best available baselines at comparable bitrate, yielding a consistently better frontier in the low-to-mid rate regime. A comparison between WaterSIC and Huffman-GPTQ on a suite of zero-shot accuracy benchmarks on Llama-3.2-1B across multiple rates is given in Table C.1 in Appendix C.

Table 5.1: Comparison of wikitext2 perplexity results on Llama3.2-1B. In each group of rows WaterSIC-FT achieves the best PPL while having minimal rate. The unquantized model perplexity is 9.76.

Method	Avg. Bitwidth	WikiText-2 PPL ↓
WaterSIC	1.00	82.44
WaterSIC-FT	**1.00**	**30.47**
WaterSIC-FT	**1.50**	**18.50**
WaterSIC	1.50	30.73
Huffman-GPTQ	1.52	1000+
Huffman-GPTQ	1.94	86.80
WaterSIC-FT	**2.00**	**13.59**
WaterSIC	2.00	16.19
QTIP	2.02	18.67
WaterSIC-FT	**2.50**	**11.33**
WaterSIC	2.50	11.83
Huffman-GPTQ	2.54	17.74
Huffman-GPTQ	2.97	13.99
WaterSIC-FT	**3.00**	**10.45**
WaterSIC	3.00	10.57
QTIP	3.02	11.17
NestQuant (W-only)	3.18	10.85
WaterSIC-FT	**3.18**	**10.27**
WaterSIC	3.18	10.35
AWQ (g128)	3.25	16.74
Huffman-GPTQ	3.45	11.65
WaterSIC-FT	**3.50**	**10.08**
WaterSIC	3.50	10.11
NestQuant (W-only)	3.50	10.38
NestQuant (W-only)	3.76	10.18
WaterSIC-FT	**3.76**	**9.98**
WaterSIC	3.76	9.99
NestQuant (W-only)	3.99	10.06
WaterSIC-FT	**4.00**	**9.91**
WaterSIC	4.00	9.92
Huffman-GPTQ	4.01	10.66
QTIP	4.02	10.12
AWQ (g128)	4.25	10.84

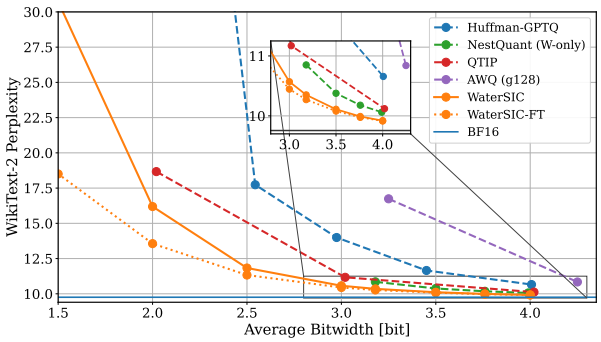

Figure 5.1: Llama-3.2-1B: WaterSIC and WaterSIC-FT vs other algorithms. WaterSIC, WaterSIC-FT and Huffman-GPTQ use entropy to report rates, others use log-cardinality.

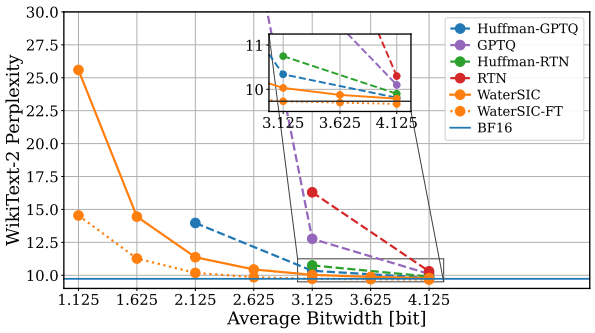

Figure 5.2: Qwen3-8B: WaterSIC and WaterSIC-FT vs other algorithms. WaterSIC, WaterSIC-FT, Huffman-GPTQ and Huffman-RTN use entropy to report rates, others use log-cardinality.

Two further views of WaterSIC on Llama-3.2-1B are informative: its joint WikiText-2 and C4 perplexity, which exposes the cost of calibrating exclusively on WikiText-2, and its KL divergence to the BF16 model, a sharper measure than perplexity of fidelity to the unquantized model.

Table 5.2 reports the WikiText-2 and C4 perplexities of WaterSIC and WaterSIC-FT across rates. WaterSIC is calibrated on WikiText-2 train at CTX=2048, and WaterSIC-FT denotes the same model after post-quantization finetuning on WikiText-2 at CTX=2048; both are evaluated at CTX=2048 on WikiText-2 test and C4 validation. Because calibration is performed exclusively on WikiText-2, the C4 gap to BF16 is wider than the W2 gap at low rates. For instance, at 1 bit WaterSIC-FT reaches W2 PPL 32.85 (23.12 above the BF16 reference of 9.73) but C4 PPL 160.70 (147.93 above the BF16 reference of 12.77), and finetuning on WikiText-2 alone does not close this gap. We examine the role of the calibration and finetuning corpus in detail in Section 5.2.

Table 5.2: Llama-3.2-1B WikiText-2 and C4 test perplexity at various rates. WaterSIC is calibrated on WikiText-2 train at CTX=2048; WaterSIC-FT is the same model after post-quantization finetuning on WikiText-2 at CTX=2048. Both evaluated at CTX=2048. Unquantized (BF16): W2 PPL 9.73, C4 PPL 12.77.

Rate	WaterSIC W2↓	WaterSIC C4↓	WaterSIC-FT W2↓	WaterSIC-FT C4↓
1.00	82.45	643.29	32.85	160.70
1.50	30.73	140.72	19.16	59.97
2.00	16.19	37.73	13.59	27.80
2.50	11.83	18.81	11.34	17.80
3.00	10.57	15.26	10.45	15.06
3.18	10.35	14.73	10.27	14.61
3.50	10.11	14.03	10.08	13.99
3.76	9.99	13.62	9.98	13.60
4.00	9.92	13.35	9.91	13.34

To complement the perplexity analysis, we also report the KL divergence between the unquantized (BF16) and quantized model output distributions, evaluated on WikiText-2 test:

$$\mathrm{KLD} \;=\; \mathrm{KL}\big(P_{\mathrm{BF16}} \,\|\, P_{\mathrm{quant}}\big) \;=\; \sum_{x} P_{\mathrm{BF16}}(x)\, \log \frac{P_{\mathrm{BF16}}(x)}{P_{\mathrm{quant}}(x)}.$$

Figure 5.3 plots this quantity against the average bitwidth for Huffman-GPTQ, WaterSIC, and WaterSIC-FT; the shaded region marks the KL gap between Huffman-GPTQ and WaterSIC at matched bitwidth. WaterSIC significantly reduces the KL divergence over Huffman-GPTQ, especially at low bitrates, and post-quantization finetuning further reduces it.

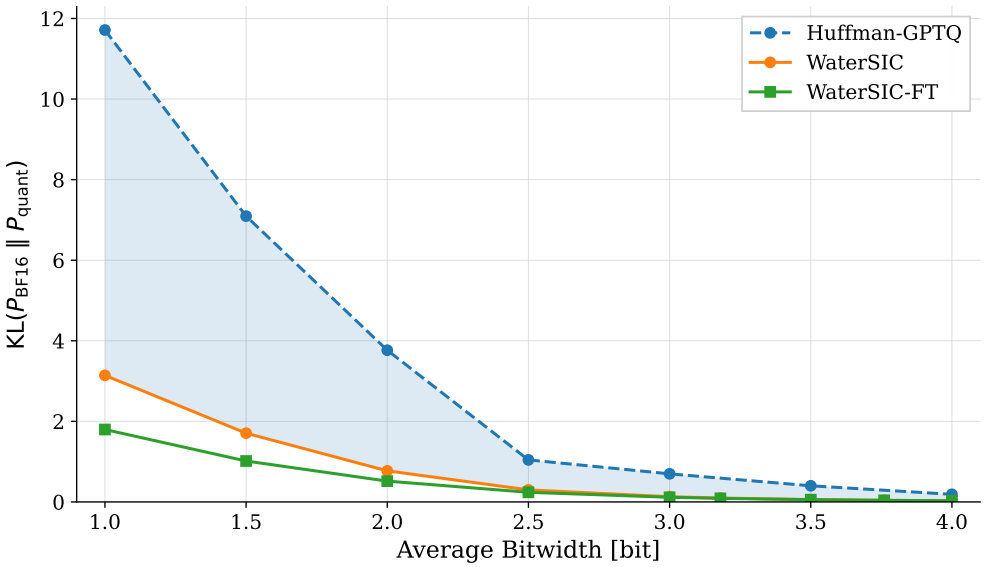

Figure 5.3: Llama-3.2-1B: KL divergence between BF16 and quantized output distributions, $\mathrm{KL}(P_{\mathrm{BF16}} \,\|\, P_{\mathrm{quant}})$, against average bitwidth, for Huffman-GPTQ, WaterSIC, and WaterSIC-FT. The shaded region marks the gap between Huffman-GPTQ and WaterSIC at matched bitwidth.

5.1.2 Qwen3-8B

Figure 5.2 and Table 5.3 summarize results on Qwen3-8B. We compare against classical weight-only baselines such as GPTQ [13] and RTN, and against entropy-based baselines (Huffman-GPTQ, Huffman-RTN) as reported and evaluated by Chen et al. [1]. WaterSIC attains state-of-the-art perplexity at all rates, demonstrating that our method performs well on a model outside of the Llama family. A comparison between WaterSIC and Huffman-GPTQ, as well as other schemes, on a suite of zero-shot accuracy benchmarks on Qwen3-8B across multiple rates is given in Table C.2 in Appendix C.

The full WikiText-2 and C4 perplexities at the same calibration / finetuning protocol are reported below.

Table 5.4 reports the WikiText-2 and C4 perplexities of WaterSIC and WaterSIC-FT across rates. The calibration, finetuning, and evaluation protocol matches Llama-3.2-1B

Method (bits)	2.125	2.625	3.125	3.625	4.125
Huffman-GPTQ	13.97	-	10.34	-	9.81
GPTQ	57.51	-	12.77	-	10.10
Huffman-RTN	593.05	-	10.75	-	9.90
RTN	2×10^{10}	-	16.30	-	10.30
WaterSIC	11.37	10.44	10.03	9.87	9.79
WaterSIC-FT	**10.18**	**9.85**	**9.73**	**9.70**	**9.67**

Table 5.3: Qwen3-8B WikiText-2 perplexity (lower is better) at fixed average bitwidths. Results for Huffman-GPTQ(HPTQ)/GPTQ/Huffman-RTN(HRTN)/RTN are taken from Chen et al. [1]; WaterSIC-FT achieves the best perplexity at all rates. The unquantized model perplexity is 9.73.

above: WaterSIC is calibrated on WikiText-2 train at CTX=2048, WaterSIC-FT is the same model after post-quantization finetuning on WikiText-2 at CTX=2048, and both are evaluated at CTX=2048 on WikiText-2 test and C4 validation.

Table 5.4: Qwen3-8B WikiText-2 and C4 test perplexity at various rates. WaterSIC is calibrated on WikiText-2 train at CTX=2048; WaterSIC-FT is the same model after post-quantization finetuning on WikiText-2 at CTX=2048. Both evaluated at CTX=2048. Unquantized (BF16): W2 PPL 9.73, C4 PPL 13.30.

Rate	WaterSIC W2↓	WaterSIC C4↓	WaterSIC-FT W2↓	WaterSIC-FT C4↓
1.12	25.59	102.52	14.54	43.17
1.62	14.45	33.96	11.27	24.33
2.12	11.37	18.64	10.18	16.54
2.62	10.44	15.15	9.85	14.50
3.12	10.03	14.11	9.73	13.83
3.62	9.87	13.77	9.70	13.63
4.12	9.79	13.57	9.67	13.49

5.1.3 Llama-3-8B

We report rate-distortion results on Llama-3-8B in Table 5.5 and Figure 5.4. The protocol matches Llama-3.2-1B and Qwen3-8B above: calibration on WikiText-2 train at CTX=2048, evaluation on WikiText-2 test at CTX=2048, and (for WaterSIC-FT) finetuning on WikiText-2 train at CTX=2048; the unquantized (BF16) reference is W2 PPL 6.14. We compare against Huffman-GPTQ [1], GPTVQ [35], NestQuant [23], and AWQ [12]. WaterSIC-FT achieves the lowest perplexity at every rate group from 1 to 4 bits, with the gap to the next-best baseline largest in the low-bit regime (e.g., 8.06 vs. Huffman-GPTQ's 22.79 at ≈1.9 bits, 7.04 vs. Huffman-GPTQ's 11.46 at ≈2.5 bits).

Table 5.5: Comparison of WikiText-2 perplexity results on Llama-3-8B, evaluated at CTX=2048. WaterSIC-FT is finetuned on WikiText-2 at CTX=2048. In each rate group, WaterSIC-FT achieves the best PPL at minimal rate. The unquantized (BF16) model perplexity is 6.14.

Method	Avg. Bitwidth	WikiText-2 PPL
WaterSIC-FT	**1.00**	**18.97**
WaterSIC	1.00	47.15
WaterSIC-FT	**1.50**	**10.90**
WaterSIC	1.50	15.89
Huffman-GPTQ	1.91	22.79
WaterSIC-FT	**2.00**	**8.06**
WaterSIC	2.00	8.93
GPTVQ	2.12	9.94
GPTVQ	2.25	9.59
WaterSIC-FT	**2.50**	**7.04**
WaterSIC	2.50	7.25
Huffman-GPTQ	2.51	11.46
Huffman-GPTQ	2.94	8.57
WaterSIC-FT	**3.00**	**6.60**
WaterSIC	3.00	6.65
GPTVQ	3.12	7.00
WaterSIC-FT	**3.18**	**6.50**
WaterSIC	3.18	6.53
NestQuant (W-only)	3.18	6.70
Huffman-GPTQ	3.41	7.36
WaterSIC-FT	**3.50**	**6.37**
WaterSIC	3.50	6.38
NestQuant (W-only)	3.50	6.49
WaterSIC-FT	**3.76**	**6.30**
WaterSIC	3.76	6.31
NestQuant (W-only)	3.76	6.38
WaterSIC-FT	**4.00**	**6.25**
WaterSIC	4.00	6.26
NestQuant (W-only)	3.99	6.31
AWQ (W4A16 g128)	4.00	6.54
Huffman-GPTQ	3.97	6.74

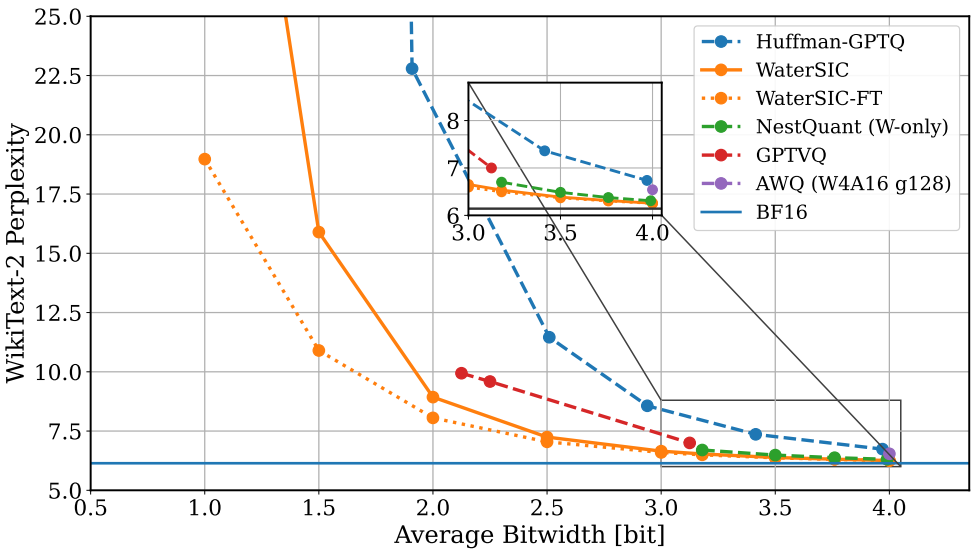

Figure 5.4: Llama-3-8B: WaterSIC vs. other algorithms. WaterSIC and Huffman-GPTQ report rates as entropy; the other methods use log-cardinality.

To compare against prior 2-bit methods we additionally re-evaluate WaterSIC at the longer context length 8192 used in those works, with finetuning likewise performed at CTX=8192. Table 5.6 reports the comparison; baseline numbers are taken from [2].

Table 5.6: Llama-3-8B 2-bit comparison, evaluated at context length 8192. WaterSIC is calibrated on WikiText-2 train at CTX=2048 and finetuned at CTX=8192. Baseline numbers from [2].

Method	Bitwidth	W2 PPL↓	C4 PPL↓
BF16	16	5.54	7.10
QuIP	2.01	76.95	98.47
DB-LLM	2.01	12.77	14.82
PV-Tuning	2.01	**6.99**	**8.29**
WaterSIC (no FT)	2.00	8.01	13.48
WaterSIC (FT W2)	2.00	7.20	11.69
WaterSIC (FT C4)	2.00	7.39	9.61
WaterSIC (FT RP)	2.00	7.35	9.60

5.1.4 Llama-2-7B

We report rate-distortion results on Llama-2-7B in Table 5.7 and Figure 5.5, again under the calibration / finetuning / evaluation protocol of the previous models (CTX=2048 throughout); the unquantized (BF16) reference is W2 PPL 5.47. We compare against Huffman-GPTQ [1], QuIP# [7], GPTVQ [35], NestQuant [23], and AWQ [12]. WaterSIC-FT again leads at every rate group, with the largest absolute improvements again concentrated in the 1-2 bit regime

where Huffman-GPTQ collapses (W2 PPL 7.64 for WaterSIC-FT at 1.5 bits, against 55.84 for Huffman-GPTQ at ≈2 bits).

Table 5.7: Comparison of WikiText-2 perplexity results on Llama-2-7B, evaluated at CTX=2048. WaterSIC-FT is finetuned on WikiText-2 at CTX=2048. In each rate group, WaterSIC-FT achieves the best PPL at minimal rate. The unquantized (BF16) model perplexity is 5.47.

Method	Avg. Bitwidth	WikiText-2 PPL
WaterSIC-FT	**1.00**	**12.03**
WaterSIC	1.00	22.43
WaterSIC-FT	**1.50**	**7.64**
WaterSIC	1.50	8.97
Huffman-GPTQ	1.53	83.23
Huffman-GPTQ	1.95	55.84
WaterSIC-FT	**2.00**	**6.29**
WaterSIC	2.00	6.57
QuIP#	2.00	6.66
GPTVQ	2.12	7.18
GPTVQ	2.25	6.99
WaterSIC-FT	**2.50**	**5.81**
WaterSIC	2.50	5.88
Huffman-GPTQ	2.55	6.38
WaterSIC-FT	**3.00**	**5.63**
WaterSIC	3.00	5.65
QuIP#	3.00	5.79
GPTVQ	3.12	5.83
Huffman-GPTQ	2.98	5.96
WaterSIC-FT	**3.50**	**5.54**
WaterSIC	3.50	5.55
Huffman-GPTQ	3.46	5.74
WaterSIC-FT	**4.00**	**5.50**
WaterSIC	4.00	5.51
NestQuant (W-only)	4.00	5.53
QuIP#	4.00	5.56
AWQ (W4A16 g128)	4.00	5.60
Huffman-GPTQ	4.01	5.62

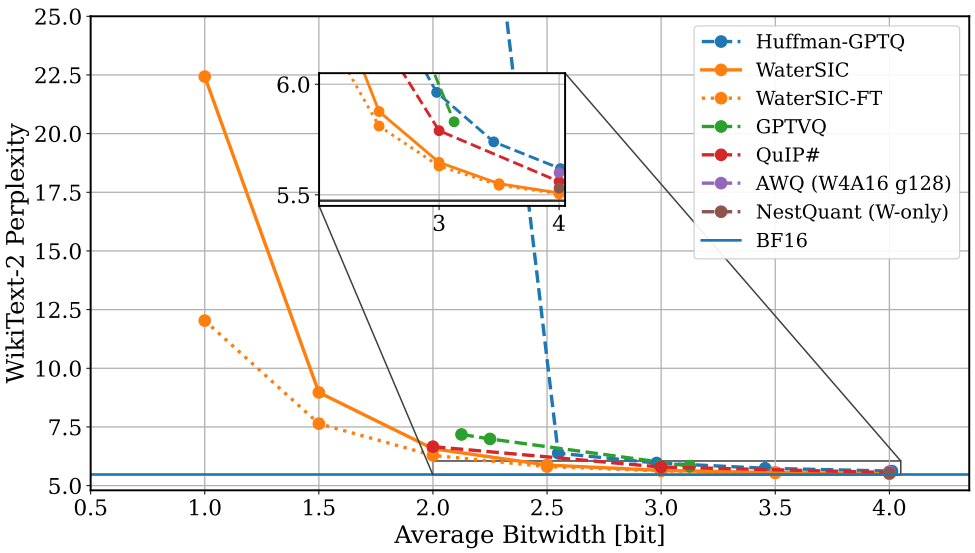

Figure 5.5: Llama-2-7B: WaterSIC vs. other algorithms. WaterSIC and Huffman-GPTQ report rates as entropy; the other methods use log-cardinality.

We additionally study the sensitivity of the algorithm to calibration context length at 2 bits, this time at the longer evaluation context CTX=4096 used in prior works. In every configuration of this experiment, finetuning is performed at the evaluation context length 4096 on the full WikiText-2 training split, so the experiment isolates the effect of the calibration context. Segmenting WikiText-2 train at non-overlapping intervals yields roughly twice as many sequences at CTX=1024 (∼2378) as at CTX=2048 (∼1189), so the shorter calibration context trades capturing of longer-range dependencies for a better-conditioned Hessian estimate. Table 5.8 reports WaterSIC at calibration contexts 1024 and 2048 across four finetuning configurations. The two calibration contexts produce nearly identical perplexities (within 0.03 on W2 and 0.05 on C4), and the two effects above appear to roughly cancel out, suggesting that the calibration context length has little effect on final quality once the model is finetuned at the evaluation context length.

Table 5.8: Llama-2-7B at 2 bits: effect of calibration context length on WaterSIC. Calibration on WikiText-2 train; finetuning on the full WikiText-2 training split segmented at the evaluation context length 4096. Evaluated at CTX=4096. Unquantized (BF16): W2 PPL 5.12, C4 PPL 6.63.

	Calib. CTX=1024		Calib. CTX=2048	
Finetuning set	W2 PPL↓	C4 PPL↓	W2 PPL↓	C4 PPL↓
None	6.152	9.853	6.123	9.813
WikiText-2	5.864	9.237	5.864	9.285
C4	5.994	8.737	5.977	8.729
RedPajama	5.986	8.832	5.976	8.818

Table 5.9 compares WaterSIC at 2 bits and CTX=4096 against AQLM, QuIP#, DB-LLM, PV-Tuning, and YAQA; baseline numbers are taken from [2] and [3]. WaterSIC's W2 PPL

matches PV-Tuning to within 0.02 (5.86 vs. 5.84 with W2 finetuning), while C4 PPL trails behind, consistent with calibrating exclusively on WikiText-2. Switching the finetuning set to C4 or RedPajama recovers most of this C4 gap: against the BF16 C4 PPL of 6.63, the gap shrinks from 2.66 ($9.29 - 6.63$, with WikiText-2 finetuning) to 2.10 ($8.73 - 6.63$, with C4 finetuning). Two of the baselines reach competitive 2-bit quality through machinery that adds inference- or training-time cost: AQLM's learned-codebook decoding requires multiple lookups and summations per dequantization, and PV-Tuning's alternating discrete-continuous optimization is substantially more expensive than the continuous-only finetuning of WaterSIC-FT. The comparable quality at 2 bits is therefore achieved with a scalar-decoded, continuous-only-finetuned codebook.

Table 5.9: Llama-2-7B 2-bit comparison, evaluated at context length 4096. WaterSIC is calibrated on WikiText-2 train at CTX=2048 and finetuned at CTX=4096. Baseline numbers from [2] and [3].

Method	Bitwidth	W2 PPL↓	C4 PPL↓
BF16	16	5.12	6.63
AQLM	2.02	6.64	8.56
QuIP#	2.01	6.19	8.16
DB-LLM	2.01	7.23	9.62
YAQA-B (INT2)	2.00	7.45	9.22
PV-Tuning	2.02	**5.84**	**7.62**
WaterSIC (no FT)	2.00	6.12	9.81
WaterSIC (FT W2)	2.00	5.86	9.29
WaterSIC (FT C4)	2.00	5.98	8.73
WaterSIC (FT RP)	2.00	5.98	8.82

5.1.5 Llama-3-70B

We report initial results on Llama-3-70B at 4 bits in Table 5.10. We ran the algorithm on 2 H100 GPUs with tensor parallelism and no post-quantization finetuning, evaluating WikiText-2 perplexity at context length 2048. WaterSIC achieves $W2\,\mathrm{PPL} = 3.06$, the lowest among the methods compared.

Table 5.10: Llama-3-70B at 4 bits: WikiText-2 perplexity at CTX=2048. WaterSIC is run on 2 H100 GPUs with tensor parallelism and no post-quantization finetuning. The unquantized (BF16) reference is W2 PPL 2.86.

Method	W2 PPL↓
BF16	2.86
RTN	3.60
QuIP	3.40
GPTQ	3.30
AWQ	3.30
OstQuant	3.19
NestQuant	3.14
WaterSIC (ours)	**3.06**

5.2 Effect of Calibration and Finetuning Set

We investigate how the choice of calibration and finetuning datasets affects downstream perplexity, focusing on Llama-3.2-1B at 2 bits. We vary the calibration set between the WikiText-2 train split and a 1T-token sample of RedPajama, using the same number of sequences in both (1189, equal to the WikiText-2 train split segmented at CTX=2048). We additionally vary the finetuning set across four options: no finetuning, WikiText-2, RedPajama, and C4, using 1189 sequences of length 2048 in each case. We evaluate both WikiText-2 test PPL and C4 test PPL at CTX=2048.

Table 5.11: Llama-3.2-1B at 2 bits: effect of the calibration and finetuning sets on WaterSIC. Calibration on WikiText-2 train or RedPajama at CTX=2048; finetuning at CTX=2048 on the indicated set. Evaluated at CTX=2048. Unquantized (BF16): W2 PPL 9.73, C4 PPL 12.77.

	Calibration on WikiText-2		Calibration on RedPajama	
Finetuning set	W2 PPL↓	C4 PPL↓	W2 PPL↓	C4 PPL↓
None	16.19	37.73	27.43	30.33
WikiText-2	13.59	27.80	12.51	17.73
RedPajama	14.52	23.78	14.52	17.40
C4	14.56	23.53	14.76	17.37

Finetuning improves both W2 and C4 PPL regardless of the calibration set. Without finetuning, calibrating on WikiText-2 gives lower W2 PPL but higher C4 PPL than calibrating on RedPajama (W2/C4 of 16.19/37.73 vs. 27.43/30.33), reflecting the calibration-evaluation distribution mismatch. Surprisingly, the lowest W2 PPL of any configuration (12.51) is reached by calibrating on RedPajama and then finetuning on WikiText-2, which beats the matched WikiText-2-calibrated, WikiText-2-finetuned setting (13.59); this suggests that the broader RedPajama calibration produces statistics that finetuning can exploit more effectively.

C4 PPL, on the other hand, is minimized by RedPajama or C4 finetuning regardless of the calibration set (17.37-17.73 when calibrated on RedPajama; 23.53-23.78 when calibrated on WikiText-2).

We extend this analysis across rates from 1 to 4 bits for the WikiText-2-calibrated Llama-3.2-1B model. Figure 5.6 shows rate-distortion curves for no finetuning, finetuning on WikiText-2, and finetuning on RedPajama, evaluated on WikiText-2 test PPL (left) and C4 test PPL (right). Table 5.12 reports the full numerical results.

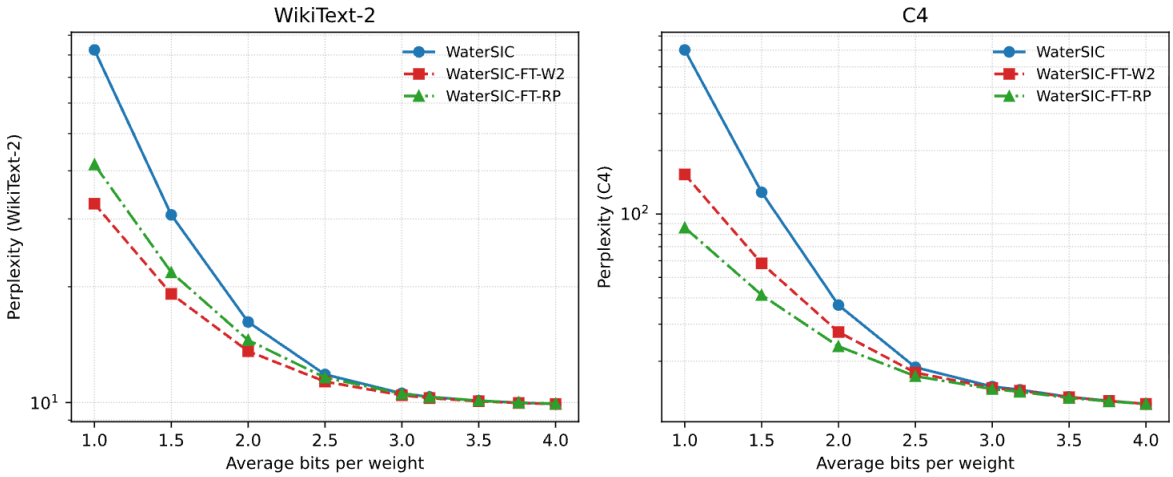

Figure 5.6: Rate-distortion curves for Llama-3.2-1B calibrated on WikiText-2, evaluated on WikiText-2 test PPL (left) and C4 test PPL (right) for three configurations: no finetuning, finetuning on WikiText-2, and finetuning on RedPajama.

Table 5.12: Llama-3.2-1B WikiText-2 and C4 test perplexity at various rates. Calibrated on WikiText-2 train at CTX=2048; finetuning at CTX=2048. Unquantized (BF16): W2 PPL 9.73, C4 PPL 12.77.

	W2 PPL↓			C4 PPL↓		
Rate	No FT	FT W2	FT RP	No FT	FT W2	FT RP
1.00	82.45	32.85	41.55	643.29	160.70	88.26
1.50	30.73	19.16	21.80	140.72	59.97	41.84
2.00	16.19	13.59	14.52	37.73	27.80	23.78
2.50	11.83	11.34	11.62	18.81	17.80	17.16
3.00	10.57	10.45	10.54	15.26	15.06	14.91
3.18	10.35	10.27	10.34	14.73	14.61	14.49
3.50	10.11	10.08	10.12	14.03	13.99	13.89
3.76	9.99	9.98	9.99	13.62	13.60	13.54
4.00	9.92	9.91	9.94	13.35	13.34	13.31

The benefit of finetuning is concentrated at low rates (1-2.5 bits). For example, at 1 bit RedPajama finetuning takes C4 PPL from 643.29 down to 88.26 (an 86% reduction) and WikiText-2 finetuning takes W2 PPL from 82.45 down to 32.85 (a 60% reduction). At rates ≥ 3.5 bits the differences between finetuning configurations shrink to within 0.15 PPL on

either evaluation, since the quantization error is by then small relative to the model's intrinsic approximation error. Comparing the two finetuning sets at low rates, WikiText-2 finetuning gives lower W2 PPL (32.85 vs. 41.55 at 1 bit) but higher C4 PPL than RedPajama finetuning (160.70 vs. 88.26 at 1 bit): the broader mixed corpus transfers better to C4, and the W2 PPL it gives up is small in absolute terms compared to the C4 PPL it recovers. This is the same domain-matching effect visible in Table 5.11: each finetuning set is best on its own evaluation distribution, and a broader finetuning corpus generalizes better off-distribution.

5.3 Limitations and Future Work

Our evaluation focuses on post-training weight quantization and does not include several complementary directions.

Loss-aware variants. We optimize Euclidean post-matmul loss layer by layer. Some of the best modern algorithms instead approximate the target PPL (or KL) loss directly, e.g. YAQA [3] and GuidedQuant [34]. WaterSIC's rate-allocation principle is independent of the choice of layerwise objective: the most concrete next step is to replace the MSE in our derivation by a YAQA-style Kronecker-factored Fisher-weighted loss, recompute the per-column rate budget under that weighted objective, and run the rest of the pipeline unchanged. We expect this to give further gains at low rates, where the KL/MSE mismatch is largest.

Joint weight-and-activation quantization. The treatment in this thesis is weight-only: input activations are assumed to be in BF16 or FP16. Our best results compose with the Qronos correction [23] (Sections 4.1 and 2.2.3), which compensates for the activation noise introduced by quantizing earlier layers, but Qronos is a calibration adjustment, not a rate-distortion-optimal scheme. To our knowledge, no analogue of Theorem 3.3 exists for the joint quantization of weights and activations. Extending the WaterSIC framework to handle quantized activations natively, with a tight characterization of the joint rate-distortion frontier and an algorithm achieving it within a constant gap, is the most consequential open problem suggested by this thesis.

End-to-end compression throughput. While we rely on entropy estimates and standard lossless codecs to translate entropy rates into compressed bitstreams, we did not benchmark end-to-end (de)compression throughput and its interaction with real hardware kernels. The practical deployability of variable-rate weight encoding (as opposed to fixed-bitwidth INT4/INT8 formats supported by current GPU kernels) is an open engineering question.

Scaling. Current experiments are conducted on relatively small to mid-sized models, with the largest being Llama-3-70B at 4 bits. Scaling to substantially larger models may expose new practical bottlenecks, both in calibration data requirements and in finetuning compute.

Appendix A

Hyperparameters

Here we summarize the hyperparameters used in our experiments and describe how each plot was produced.

Common settings. We quantized layers sequentially, collecting statistics of $(X, \hat{X}, R, \hat{R})$ at the input to each subsequent layer. The reported rate is the parameter-count-weighted average of the per-layer rates.

For evaluation, we used the test split of WikiText-2 and computed perplexity (PPL) with a context window of 2048. In addition to PPL, we computed KL divergence and several reasoning benchmarks, including MMLU and HellaSwag.

We evaluated WaterSIC and GPTQ on Llama-3.2-1B, Qwen3-8B, Llama-3-8B, Llama-2-7B and Llama-3-70B.

GPTQ. We used $\delta = 0.1$ (the default) for all models except Llama-3-8B, where we found that $\delta = 0.01$ performed significantly better at some rates. We evaluated with `groupsize` $= -1$, `blocksize` $= 128$, and `actorder` $=$ `False`. To obtain different target rates, we varied the `maxq` parameter and computed the entropy of the resulting integer matrices.

As input to the algorithm, we used quantized activation statistics $\hat{X}$, as they consistently produced better results. In the main plots, this variant is labeled as Huffman-GPTQ (HPTQ), following [1]. Entries labeled GPTQ, on the other hand, correspond to the same algorithm with the rate set to $\log_2(\texttt{maxq})$ (i.e., without entropy coding). For Qwen3-8B, we did not run Huffman-GPTQ ourselves; instead, we report the results from [1].

WaterSIC. With dead-feature erasure enabled, we found that very small damping ($\delta = 10^{-4}$) is sufficient, in contrast to the much larger damping (e.g., $\approx 10\%$) commonly used in standard GPTQ.

For each layer, we use a secant method to find the value of c in Algorithm 3 that yields the target rate. We exploit the approximately linear relationship between c and the resulting entropy: an initial evaluation at the target rate reveals an offset, and a secant correction refines the estimate in 2-3 iterations to within < 0.005 bits of the target. To reduce computation, this search compresses only a randomly sampled 10% of the rows of W. After convergence, we rerun the algorithm with the selected value of c on the full matrix to produce the final quantized weights.

As we quantize the model sequentially, we maintain a running rate budget initialized to the global target. At each step, we allocate this remaining budget evenly across the unquantized

matrices and calibrate the current layer to match its assigned share. This procedure keeps the final average rate extremely close to the requested target. In addition, because dead-feature erasure reduces the effective dimensionality (and thus the rate) of some early layers, the leftover budget is redistributed to later layers, resulting in a mild increase in per-layer rates toward the end of the network.

Our best results across models used activation drift correction (4.2) and residual stream compensation (4.4), together with weighted calibration (4.5) and adaptive mixing (4.6) for joint quantization of the (w_q, w_k, w_v) projections. We found that 10 iterations of golden-section search provide sufficient precision for each mixing parameter. The per-layer procedure is:

1. **Drift-mixing optimization.** Holding $\epsilon_{\text{aw}} = 0$ fixed, run golden-section search over $\epsilon_{\text{qr}} \in [0, 1]$. At each candidate value, re-quantize (w_q, w_k, w_v) jointly (each time finding the appropriate c via the secant method to match the target rate) and evaluate the relative MSE at the w_o input over the calibration set. Select $\epsilon^\star_{\text{qr}}$ minimizing (4.9).

2. **Attention-weighting optimization.** Fixing $\epsilon_{\text{qr}} = \epsilon^\star_{\text{qr}}$, run golden-section search over $\epsilon_{\text{aw}} \in [0, 1]$ using the same re-quantize-and-evaluate procedure, yielding $\epsilon^\star_{\text{aw}}$.

3. **Final quantization.** With $(\epsilon^\star_{\text{qr}}, \epsilon^\star_{\text{aw}})$ fixed, quantize (w_q, w_k, w_v) at the target rate assigned by the global rate controller.

Each golden-section iteration requantizes three matrices for a single layer (with the secant method determining the scale c at each evaluation point) and runs a forward pass through the attention block on the calibration set. Because the secant method converges in 2-3 iterations, rate-matching each evaluation point adds minimal overhead compared to the quantization itself.

Qwen3-8B. For this model, we identified a small number of outlier rows in the FFN gate and up-projection matrices that negatively impact quantization stability. Specifically, rows 5723 and 8518 in layer 6 (w_1, w_3) and rows 2271 and 1875 in layer 16 (w_1, w_3) have anomalously large norms. These outliers inflate the effective scale of the corresponding down-projection w_2 (which must compensate for their magnitude), leading to degraded quantization fidelity. We verified that zeroing these rows in the *unquantized* model increases WikiText-2 test perplexity by less than 0.01, indicating that they encode near-degenerate features rather than useful signal. We therefore zero these rows in both the unquantized and quantized models prior to quantization and, correspondingly, zero the associated columns in w_2. This removes a source of numerical instability without a meaningful loss in model quality, and substantially reduces quantization error in the affected layers.

Additionally, on Qwen3-8B we observed a sharp transition around layer 16: the relative MSE at the input to the layer-16 w_2 matrix is substantially higher than in earlier layers. After this point, the coordinate search increasingly prefers the unquantized statistics over the quantized-model ones: as shown in Table A.1, the optimal drift-mixing coefficients $\epsilon^\star_{\text{qr}}$ are close to 1 for most subsequent layers (across all three rates). In contrast, the attention-weight mixing parameter $\epsilon^\star_{\text{aw}}$ remains at 0 (full attention weighting) for the majority of layers and becomes nonzero only in a subset of deeper layers (Table A.2). We hypothesize that beyond this depth, quantization noise is strongly amplified, which degrades the quality of the

estimated quantized-model covariances and makes activation drift compensation less effective in practice.

Table A.1: Optimal drift-mixing coefficients $\epsilon^\star_{\text{qr}}$ selected by adaptive mixing for Qwen3-8B. Values are reported per layer for three target rates (bits per parameter). Larger $\epsilon^\star_{\text{qr}}$ indicates a stronger preference for the unquantized statistics Σ_X over the drift-corrected statistics $(\Sigma_{\hat{X}}, \Sigma_{X,\hat{X}})$. Note a "phase change" at layer 15.

Layer	2.125	3.125	4.125	Layer	2.125	3.125	4.125
0	0.7641	0.5064	1.0000	18	0.9784	1.0000	0.9703
1	0.0132	0	0	19	1.0000	0.9786	0.9913
2	0.0132	0.0016	0	20	1.0000	1.0000	1.0000
3	0	0.0201	0.0098	21	0.9657	0.9874	1.0000
4	0.1459	0	0.0081	22	0.9956	1.0000	1.0000
5	0.0545	0	0.0047	23	1.0000	0.9987	1.0000
6	0	0.0473	0.0900	24	0.9951	0.9917	1.0000
7	0.2353	0.2905	0.1459	25	0.9894	0.9951	0.9951
8	0.1462	0.3824	0	26	0.9820	0.9874	0.9542
9	0.2844	0.0803	0.2937	27	0.9874	0.9820	0.9870
10	0.3262	0.0701	0.1653	28	0.9561	1.0000	0.9755
11	0	0	0	29	0.9166	0.9820	0.9738
12	0.3820	0.1834	0.2918	30	0.7639	1.0000	0.9868
13	0.8204	0.3851	0.4719	31	1.0000	1.0000	0.9855
14	0.7685	0.2535	0.4427	32	1.0000	0.9525	0.9836
15	0.9849	0.9392	0.9230	33	1.0000	0.9917	1.0000
16	0.9950	1.0000	0.9098	34	1.0000	1.0000	1.0000
17	0.9675	1.0000	0.9769	35	0.8198	0.8404	0.8968

Table A.2: Optimal attention-weight mixing coefficients $\epsilon^\star_{\text{aw}}$ selected by adaptive mixing for Qwen3-8B. Values are reported per layer for three target rates (bits per parameter). Here $\epsilon^\star_{\text{aw}} = 0$ corresponds to full attention-weighted calibration, while larger values interpolate toward the uniformly weighted covariances.

Layer	2.125	3.125	4.125	Layer	2.125	3.125	4.125
0	0.5197	0.4721	0.5492	18	0	0	0
1	0	0	0	19	0.3839	0.6180	0.4114
2	0	0	0	20	0	0	0
3	0	0	0	21	1.0000	0.4752	0.2012
4	0	0	0	22	0.2310	0.4191	0.4047
5	0	0	0	23	0.2710	1.0000	0.3731
6	0	0	0	24	0.4346	0.2341	0.4741
7	0	0	0	25	0.3951	0.3820	0.7082
8	0	0	0	26	0.4769	0.3212	0.4702
9	0	0	0	27	0.3293	0.3820	0.7639
10	0	0	0	28	0.3607	0.7082	1.0000
11	0	0	0	29	0.5279	0.3731	0.6130
12	0	0	0	30	0	0	0
13	0	0	0	31	1.0000	0.3839	0.6180
14	0	0	0	32	0.3769	0.6161	0.0914
15	0	0	0	33	0.8448	0.6262	0.5798
16	0	0	0	34	0	0.2361	0.6130
17	0	0	0	35	0.0007	0.0027	0.0024

WaterSIC-FT. For post-quantization finetuning, we use AdamW without weight decay, with cosine annealing from a peak learning rate of 5×10^{-4} to 5×10^{-6} over 4 epochs on the full WikiText-2 training split (1189 sequences of length 2048). Gradient checkpointing is applied at every transformer block to keep activation memory at $O(1)$ blocks. The teacher's final hidden states (before the output head) are precomputed once and cached on CPU, eliminating the need to run the teacher forward pass at each training step and roughly halving per-step compute. During the forward pass, integer codes Z are loaded from CPU to GPU per-layer on demand, so that peak GPU memory is dominated by a single block's activations rather than the full quantized model. The rescaler parameters t and γ are rounded to half-precision via a straight-through estimator during training, so the optimizer adapts to the deployed (BF16-stored) parameter precision.

Appendix B

Diagnostic Plots and Ablations

Row-column rescalers. On Fig. B.1 we demonstrate how values of T (row rescalers) and Γ (column rescalers) change with rate. As expected theoretically, the LMMSE correction is indispensable for low-rate quantization and is not necessary for rates above $R \geq 4$ bit.

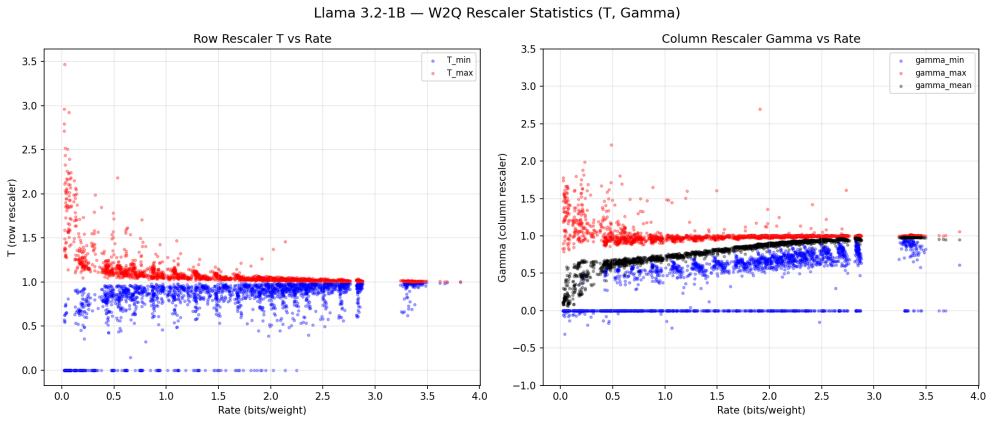

Figure B.1: Diagonal rescalers statistics. Llama-3.2-1B, various rates. Row is the reduction (in-channel) dimension, so that row rescaler affects one out-channel, and column rescaler affects one in-channel.

Zero coordinates and dead features. One issue we encountered is that LayerNorm in Llama-3.2-1B effectively zeros out certain coordinates at layer inputs, which can make the empirical covariance matrices nearly singular. See Table B.1 for the list of offending coordinates, defined as those with variance $< 10^{-4}$ times the mean variance across all 2048 coordinates. This observation motivated the *dead feature erasure* heuristic described in Section 4: we remove near-zero-variance input dimensions before computing the Cholesky factor and optimizing rescalers, solve the reduced system, and then expand the quantized weight back to the original size by inserting zeros at the erased positions. In practice, this greatly improves numerical stability (both for the Cholesky decomposition and for rescaler optimization) while having negligible effect on reconstruction quality, since the erased coordinates carry little signal.

Layer	Low-variance input indices
Layer 0, ATTN input	278*
Layer 0, MLP input	64*, 146*, 509, 1280*, 1618*, 1938*, 2023*
Layer 1, ATTN input	146, 735, 762, 841, 894, 1002, 1314, 1334, 1476, 1503, 1619, 2037
Layer 1, MLP input	64*, 146*, 509, 1228, 2023
Layer 2, ATTN input	1159, 1314, 2023*
Layer 2, MLP input	146*, 2023*
Layer 3, MLP input	146, 2023
Layer 4, MLP input	146
Layer 5, MLP input	146

Table B.1: Llama-3.2-1B: input features with standard deviation below 1% (asterisk) and 0.1% (no asterisk) of the mean standard deviation among all coordinates. These near-zero coordinates are produced by a diagonal multiplier inside the RMSnorm at the respective layer's input.

Unequal rate. Fig. B.2 shows an example of the rate distribution among the different columns (in-channels) of matrices in one particular layer, as well the rate distribution among all the columns (of all layers) of Qwen3-8B, when compressing to target rate of 2.0 bits.

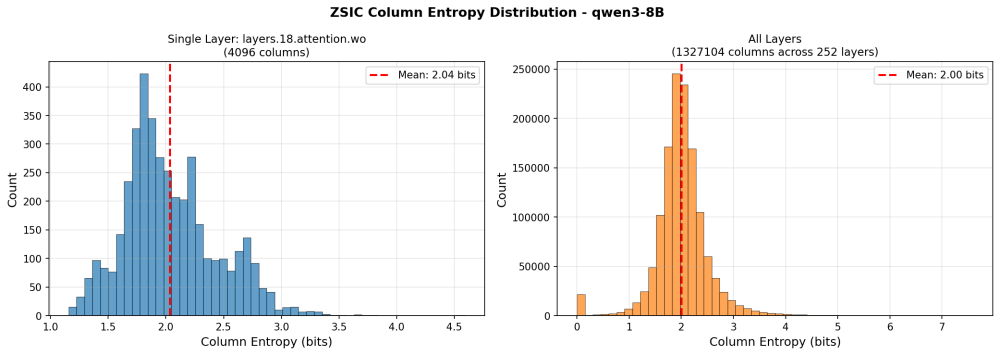

Figure B.2: Distribution of actual compression rates per in-channel inside a single layer (left) and over all layers (right) for Qwen3-8B model.

Additionally, we study how well the entropy translates into algorithmically achievable compression. For each quantized weight matrix, we serialize the integer codes column-by-column (i.e., all entries sharing the same input feature are contiguous in the byte stream) and pack them into the smallest sufficient integer type (`int8` or `int16`). We then compress this byte stream with standard lossless codecs, including Zstandard (level 22) and LZMA (preset 9).

To this end, we evaluate several quantized layers of Llama-3.2-1B produced by our algorithm at a target rate of 2 bits per parameter. We report the entropy of all matrix entries, the maximum and average per-column entropies, along with the compression rates achieved by Zstandard and LZMA, in Table B.2.

Layer	Matrix	entropy (all matrix entries)	max(col-entropy)	avg(col-entropy)	zstd (bpp)	lzm
6	attention.wk	1.963	4.100	1.866	2.037	2
	attention.wo	1.975	4.397	1.853	2.016	2
	attention.wq	1.994	4.330	1.923	2.061	2
	attention.wv	1.979	3.355	1.922	2.049	2
	feed_forward.w1	1.993	3.869	1.938	2.045	2
	feed_forward.w2	1.986	4.162	1.892	2.094	2
	feed_forward.w3	1.994	3.451	1.958	2.066	2
7	attention.wk	1.968	4.518	1.878	2.037	2
	attention.wo	1.991	3.691	1.859	2.005	2
	attention.wq	1.984	4.400	1.924	2.035	2
	attention.wv	1.978	3.362	1.920	2.044	2
	feed_forward.w1	1.987	3.779	1.931	2.038	2
	feed_forward.w2	1.992	3.977	1.886	2.138	2
	feed_forward.w3	1.992	3.383	1.954	2.064	2

Table B.2: Per-matrix entropy statistics and compression rates (bits/parameter) for Zstandard and LZMA on `int8`-packed weights.

Ablation of individual components. We now present an ablation study illustrating the contribution of each technique described in Section 4. All plots report the relative MSE at the *input* to each layer's weight matrix, comparing two quantized models against a shared unquantized reference. Filled markers denote the first configuration (run A), and hollow markers denote the second (run B).

Residual stream correction. Figure B.3 compares WaterSIC with and without residual stream compensation (4.4) on Llama-3.2-1B at 4 bit. The improvement is concentrated at the down-projection layers (w_o and w_2), as expected: these are the only layers whose objective changes under residual compensation, since they are the only ones that contribute directly to the residual stream.

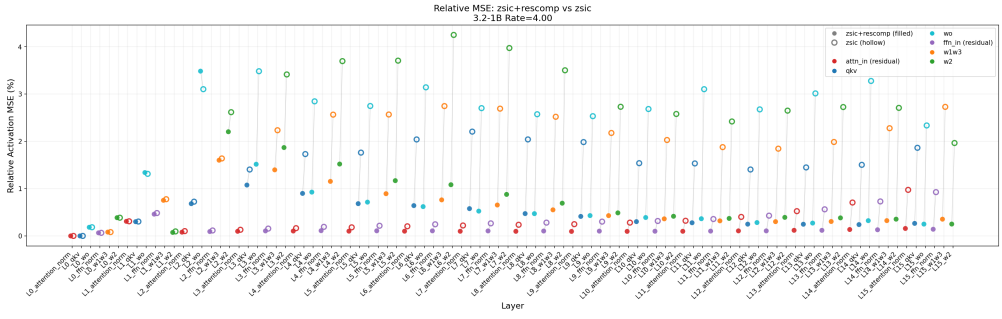

Figure B.3: Effect of residual stream compensation on Llama-3.2-1B (4 bit). Residual compensation reduces activation MSE primarily at w_o and w_2 inputs, with improvements propagating to subsequent layers through the residual stream.

Activation drift correction. Figure B.4 adds activation drift correction (Qronos) (4.3) on top of residual compensation. The drift-corrected statistics improve most layers, but notably

increase the relative MSE at w_o inputs in several layers. This occurs because the softmax nonlinearity amplifies small errors in the QKV projections: when the drift-corrected Hessian is slightly misspecified, the resulting quantization errors in w_q, w_k, w_v are magnified through attention, producing worse w_o input distortion than using the standard Hessian.

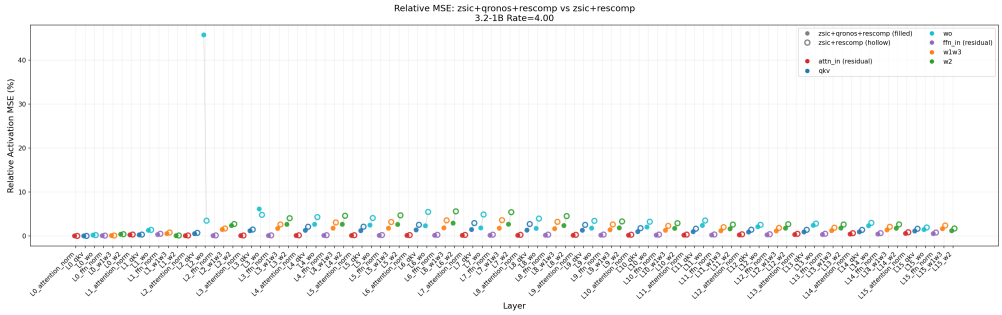

Figure B.4: Effect of activation drift correction on Llama-3.2-1B (4 bit). Adding Qronos improves most layers but degrades w_o inputs in some layers, where softmax amplifies QKV quantization errors.

Attention-weighted calibration. Figure B.5 shows that adding attention-weighted calibration (4.5) with joint adaptive mixing (4.6) resolves the w_o degradation introduced by Qronos. By reweighting the QKV covariance estimates to account for the attention sink at position 0 and jointly optimizing the mixing parameters to minimize w_o input MSE (4.9), the quantizer produces QKV weights whose errors are no longer amplified through softmax.

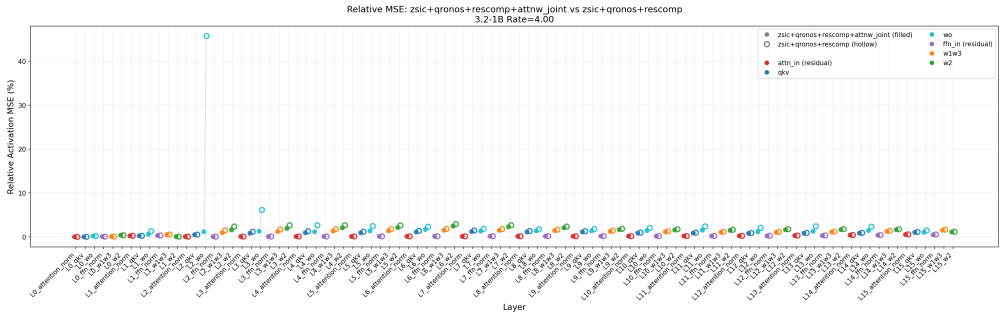

Figure B.5: Effect of attention-weighted calibration on Llama-3.2-1B (4 bit). Joint adaptive mixing with attention-weighted covariances eliminates the w_o degradation caused by Qronos.

Adaptive mixing for larger models. While the combination of Qronos, residual compensation, and attention weighting works well on smaller models, we found that activation drift correction becomes increasingly unstable in deeper layers of larger models, where $\hat{X}$ drifts further from X. Figure B.6 demonstrates this on Qwen3-8B (36 layers): the adaptive mixing procedure from (4.6) stabilizes Qronos by interpolating toward the original statistics when the drift-corrected Hessian is ill-conditioned, yielding consistent improvements across layers.

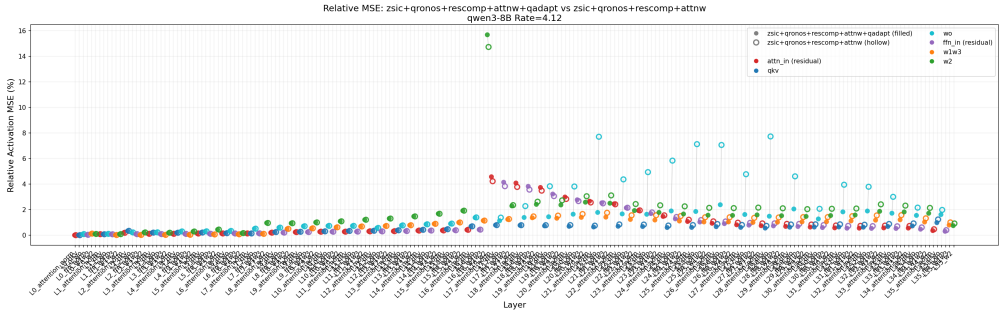

Figure B.6: Effect of adaptive mixing on Qwen3-8B (4.12 bit). Adaptive ϵ_{qr} blending stabilizes Qronos in deeper layers where pure drift correction degrades.

Cumulative effect. Finally, Figure B.7 compares the full pipeline (WaterSIC with residual compensation, Qronos, attention weighting, and adaptive mixing) against base WaterSIC on Llama-3.2-1B. The cumulative improvement is substantial and consistent across all layer types, with the largest gains at w_o and w_2 inputs, where the individual contributions compound.

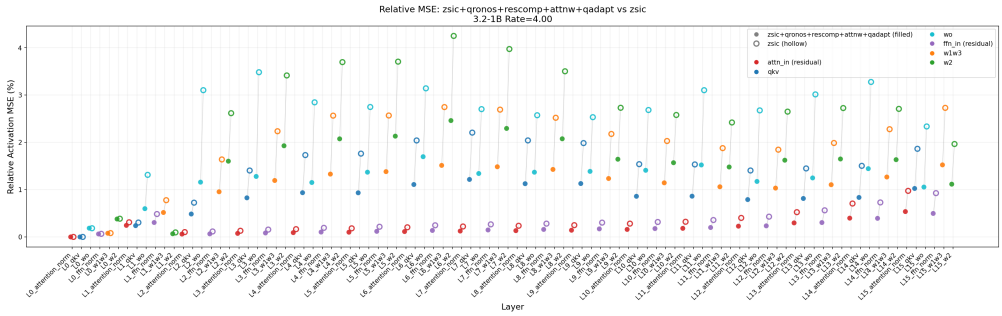

Figure B.7: Full WaterSIC pipeline vs. base WaterSIC on Llama-3.2-1B (4 bit). All techniques combined yield a consistent reduction in activation MSE across every layer.

Gaussianity of weights. Our theoretical analysis assumes i.i.d. Gaussian rows of W. We expect this to be representative of modern LLMs for two reasons. First, the random Hadamard transform applied during incoherence processing Gaussianizes weight outliers, since the resulting entries are linear combinations of $\Theta(n)$ original entries weighted by random signs. Second, the outlier features common in early LLMs have largely disappeared in models trained with modern optimizers, layer-norm placements, and learning rate schedules [28]. To verify this empirically, we compute, for each weight matrix in Llama-2-7B, the Kolmogorov-Smirnov distance between the empirical CDF and its best-fit Gaussian and Laplace CDFs, and report the per-layer-type averages over all 32 decoder layers in Table B.3.

Table B.3: Kolmogorov-Smirnov distance to the best-fit Gaussian and Laplace distributions, averaged over the 32 decoder layers of Llama-2-7B. The rightmost column reports the number of layers (out of 32) for which the Gaussian fit is closer than the Laplace fit.

Weight type	Mean $D_{\text{Gauss}} \downarrow$	Mean $D_{\text{Laplace}} \downarrow$	Gauss better
w_1	0.63%	4.13%	32/32
w_2	0.32%	4.24%	32/32
w_3	0.33%	4.26%	32/32
w_q	2.96%	2.99%	23/32
w_k	3.16%	2.66%	16/32
w_v	0.84%	3.93%	30/32
w_o	0.93%	3.99%	30/32

The FFN matrices (w_1, w_2, w_3) together with the attention value and output projections (w_v, w_o) all admit a near-perfect Gaussian fit, with $D_{\text{Gauss}} < 1\%$ in every layer and the Gaussian preferred over Laplace in $\geq 30/32$ layers per type. The query and key projections (w_q, w_k) deviate the most: their tails are slightly heavier, the Gaussian and Laplace fits are within $\sim 0.3\%$ of each other on average, and the Laplace fit is preferred in roughly half of the layers for w_k. We observe the same pattern on Llama-3-8B and Llama-3.2-1B: the FFN matrices are consistently Gaussian, and w_q, w_k are the only matrices for which a Laplace fit is occasionally closer. Figure B.8 shows representative weight histograms with best-fit Gaussian and Laplace overlays for several layers of Llama-3.2-1B.

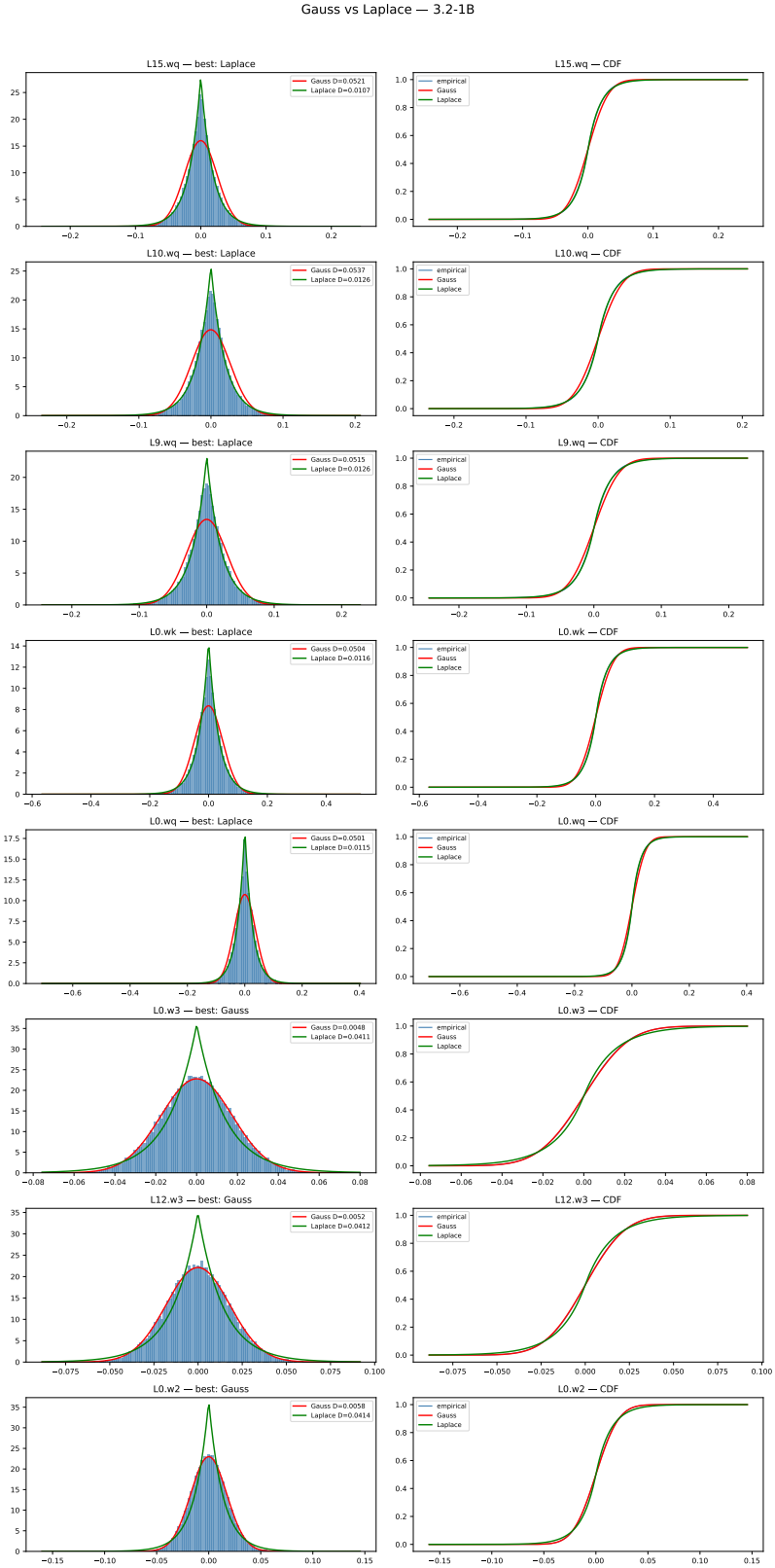

Figure B.8: Weight histograms for selected layers of Llama-3.2-1B with best-fit Gaussian (blue) and Laplace (orange) overlays.

Appendix C

Zero-Shot Accuracy Evaluations

The previous sections focused on perplexity on WikiText-2 and C4. We now turn to downstream zero-shot accuracy across a range of standard benchmarks, to verify that the perplexity advantages of WaterSIC translate into corresponding gains on tasks the quantized model is actually deployed for.

Table C.1 compares WaterSIC and HPTQ on Llama-3.2-1B across bitrates. At every rate, WaterSIC outperforms HPTQ on the majority of benchmarks and remains competitive on the rest. Table C.2 compares WaterSIC against the Qwen3-8B numbers reported in [1]; the same pattern holds, with WaterSIC matching or improving over the reported HPTQ numbers on most benchmarks.

Rate	Method	ARC-C	ARC-E	HellaSwag	OBQA	PIQA	SocialIQA	Wino
1.50	Huffman-GPTQ	**0.2722**	0.2597	0.2619	**0.2960**	0.5185	0.3306	0.5067
	WaterSIC	0.2415	**0.3620**	**0.3127**	0.2720	**0.5729**	**0.3501**	**0.5107**
2.00	Huffman-GPTQ	0.2218	0.2959	0.3030	0.2520	0.5332	0.3465	0.5075
	WaterSIC	**0.2679**	**0.4655**	**0.4052**	**0.2920**	**0.6066**	**0.3685**	**0.5328**
2.50	Huffman-GPTQ	0.2534	0.4327	0.4215	0.2880	0.6132	0.3593	0.5454
	WaterSIC	**0.3166**	**0.5551**	**0.5205**	**0.3400**	**0.6774**	**0.3992**	**0.5912**
3.00	Huffman-GPTQ	0.3038	0.5215	0.5125	0.2960	0.6708	0.4028	0.5706
	WaterSIC	**0.3387**	**0.5720**	**0.5872**	**0.3580**	**0.7182**	**0.4248**	**0.5991**
3.50	Huffman-GPTQ	0.3217	0.5606	0.5613	0.3320	0.7133	0.4161	0.5872
	WaterSIC	**0.3601**	**0.6162**	**0.6166**	**0.3700**	**0.7421**	**0.4217**	**0.6140**
4.00	Huffman-GPTQ	0.3447	0.6040	0.6098	0.3600	0.7291	0.4212	0.6054
	WaterSIC	**0.3737**	**0.6170**	**0.6287**	**0.3800**	**0.7497**	**0.4340**	**0.6069**

Table C.1: Zero-shot accuracy on Llama-3.2-1B. For each rate and task, we bold the better result between Huffman-GPTQ and WaterSIC (higher is better).

Rate	Method	Wino	MMLU	HSwag	PIQA	SciQ	TQA-MC1	TQA-MC2
16.000	BF16	68.11	73.02	74.90	77.80	95.70	36.35	54.50
4.125	Huffman-GPTQ	67.17	72.28	**74.84**	77.42	95.60	35.01	53.36
	GPTQ	68.82	71.76	74.22	**77.58**	95.30	36.35	54.55
	HRTN	67.56	72.15	74.72	76.99	94.20	36.47	**56.46**
	RTN	67.17	69.71	74.30	75.90	94.50	**36.84**	55.77
	WaterSIC	**70.09**	**72.77**	74.60	76.99	**95.90**	35.99	53.80
3.125	Huffman-GPTQ	66.93	**70.96**	73.18	**77.53**	**95.40**	36.11	54.73
	GPTQ	68.35	65.80	70.80	75.46	75.46	36.11	**55.21**
	HRTN	66.22	67.85	72.36	76.12	93.70	35.13	53.68
	RTN	57.93	47.90	55.41	70.89	87.10	34.03	52.76
	WaterSIC	**68.43**	70.53	**73.38**	76.99	94.70	**36.60**	53.95
2.125	Huffman-GPTQ	59.19	52.99	63.86	72.52	86.80	31.09	49.01
	GPTQ	52.25	34.25	39.32	57.83	57.83	28.40	46.91
	HRTN	51.22	33.91	49.27	65.78	76.80	30.48	**51.78**
	RTN	49.08	22.95	26.04	51.63	21.20	24.11	47.33
	WaterSIC	**67.01**	**61.14**	**64.52**	**74.05**	**92.60**	**32.80**	49.11

Table C.2: Zero-shot accuracy on Qwen3-8B. Bold indicates the best value within each rate block for each benchmark (higher is better).

References

[1] J. Chen, Y. Shabanzadeh, E. Crnčević, T. Hoefler, and D. Alistarh. "The Geometry of LLM Quantization: GPTQ as Babai's Nearest Plane Algorithm." *arXiv preprint arXiv:2507.18553*, 2025.

[2] V. Malinovskii, D. Mazur, I. Ilin, D. Kuznedelev, K. Burlachenko, K. Yi, D. Alistarh, and P. Richtarik. "PV-Tuning: Beyond Straight-Through Estimation for Extreme LLM Compression." *Advances in Neural Information Processing Systems*, 2024.

[3] A. Tseng, Z. Sun, and C. D. Sa. *Model-Preserving Adaptive Rounding*. 2025. arXiv: 2505.22988 [cs.LG]. URL: https://arxiv.org/abs/2505.22988.

[4] S. Savkin. "Quantization Methods for Matrix Multiplication and Efficient Transformers." M.Eng. Thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, Sept. 2025.

[5] E. Frantar, R. L. Castro, J. Chen, T. Hoefler, and D. Alistarh. "MARLIN: Mixed-Precision Auto-Regressive Parallel Inference of Large Language Models." In: *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP)*. 2025.

[6] H. Badri and A. Shaji. *Half-Quadratic Quantization of Large Machine Learning Models*. Nov. 2023. URL: https://mobiusml.github.io/hqq_blog/.

[7] A. Tseng, J. Chee, Q. Sun, V. Kuleshov, and C. D. Sa. *QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks*. 2024. arXiv: 2402.04396 [cs.LG]. URL: https://arxiv.org/abs/2402.04396.

[8] V. Egiazarian, A. Panferov, D. Kuznedelev, E. Frantar, A. Babenko, and D. Alistarh. *Extreme Compression of Large Language Models via Additive Quantization*. 2024. arXiv: 2401.06118 [cs.LG]. URL: https://arxiv.org/abs/2401.06118.

[9] A. Tseng, Q. Sun, D. Hou, and C. D. Sa. *QTIP: Quantization with Trellises and Incoherence Processing*. 2025. arXiv: 2406.11235 [cs.LG]. URL: https://arxiv.org/abs/2406.11235.

[10] A. Panferov, J. Chen, S. Tabesh, R. L. Castro, M. Nikdan, and D. Alistarh. "QuEST: Stable Training of LLMs with 1-Bit Weights and Activations." *arXiv preprint arXiv:2502.05003*, 2025.

[11] J. Cook, J. Guo, G. Xiao, Y. Lin, and S. Han. "Four Over Six: More Accurate NVFP4 Quantization with Adaptive Block Scaling." *arXiv preprint arXiv:2512.02010*, 2025.

[12] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration*. 2024. arXiv: 2306.00978 [cs.CL]. URL: https://arxiv.org/abs/2306.00978.

[13] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. 2023. arXiv: 2210.17323 [cs.LG]. URL: https://arxiv.org/abs/2210.17323.

[14] J. Chee, Y. Cai, V. Kuleshov, and C. D. Sa. *QuIP: 2-Bit Quantization of Large Language Models With Guarantees*. 2024. arXiv: 2307.13304 [cs.LG]. URL: https://arxiv.org/abs/2307.13304.

[15] Z. Liu, C. Zhao, H. Huang, S. Chen, J. Zhang, J. Zhao, S. Roy, L. Jin, Y. Xiong, Y. Shi, et al. "Paretoq: Scaling laws in extremely low-bit llm quantization." *arXiv preprint arXiv:2502.02631*, 2025.

[16] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo. "OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models." In: *International Conference on Learning Representations*. 2024.

[17] F. Cheng, C. Guo, C. Wei, J. Zhang, C. Zhou, E. Hanson, J. Zhang, X. Liu, H. Li, and Y. Chen. "Ecco: Improving Memory Bandwidth and Capacity for LLMs via Entropy-aware Cache Compression." In: *International Symposium on Computer Architecture (ISCA)*. 2025.

[18] P. Yubeaton et al. "Huff-LLM: End-to-End Lossless Compression for Efficient LLM Inference." *arXiv preprint arXiv:2502.00922*, 2025.

[19] P. Putzky, M. Genzel, M. Mollenhauer, S. Schulze, T. Wollmann, and S. Dietzel. *Float8@2bits: Entropy Coding Enables Data-Free Model Compression*. 2026. arXiv: 2601.22787 [cs.LG].

[20] A. Jarmusch and S. Chandrasekaran. *Microbenchmarking NVIDIA's Blackwell Architecture: An in-depth Architectural Analysis*. 2025. arXiv: 2512.02189 [cs.AR].

[21] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. "QLoRA: Efficient Finetuning of Quantized LLMs." In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[22] M. Elhoushi and J. Johnson. *any4: Learned 4-bit Numeric Representation for LLMs*. 2025. arXiv: 2507.04610 [cs.LG]. URL: https://arxiv.org/abs/2507.04610.

[23] S. Savkin, E. Porat, O. Ordentlich, and Y. Polyanskiy. *NestQuant: Nested Lattice Quantization for Matrix Products and LLMs*. 2025. arXiv: 2502.09720 [cs.LG]. URL: https://arxiv.org/abs/2502.09720.

[24] Y. Polyanskiy and Y. Wu. *Information theory: From coding to learning*. Cambridge university press, 2024.

[25] T. Linder and R. Zamir. "On the asymptotic tightness of the Shannon lower bound." *IEEE Transactions on Information Theory*, **40**(6), 1994, pp. 2026–2031.

[26] S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman. *QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs*. 2024. arXiv: 2404.00456 [cs.LG]. URL: https://arxiv.org/abs/2404.00456.

[27] Z. Liu, C. Zhao, I. Fedorov, B. Soran, D. Choudhary, R. Krishnamoorthi, V. Chandra, Y. Tian, and T. Blankevoort. *SpinQuant: LLM quantization with learned rotations*. 2025. arXiv: 2405.16406 [cs.LG]. URL: https://arxiv.org/abs/2405.16406.

[28] J. Park, T. Lee, C. Yoon, H. Hwang, and J. Kang. "Outlier-Safe Pre-Training for Robust 4-Bit Quantization of Large Language Models." *arXiv preprint arXiv:2506.19697*, 2025.

[29] J. Birnick. "The Lattice Geometry of Neural Network Quantization–A Short Equivalence Proof of GPTQ and Babai's algorithm." *arXiv preprint arXiv:2508.01077*, 2025.

[30] S. Zhang, H. Zhang, I. Colbert, and R. Saab. "Qronos: Correcting the Past by Shaping the Future... in Post-Training Quantization." *arXiv preprint arXiv:2505.11695*, 2025.

[31] H. Zhang, S. Zhang, I. Colbert, and R. Saab. "Provable post-training quantization: Theoretical analysis of optq and qronos." *arXiv preprint arXiv:2508.04853*, 2025.

[32] Y. Sun et al. *FlatQuant: Flatness Matters for LLM Quantization*. 2025. arXiv: 2410.09426 [cs.CL]. URL: https://arxiv.org/abs/2410.09426.

[33] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models*. 2024. arXiv: 2211.10438 [cs.CL]. URL: https://arxiv.org/abs/2211.10438.

[34] J. Kim, M. E. Halabi, W. Park, C. J. Schaefer, D. Lee, Y. Park, J. W. Lee, and H. O. Song. *GuidedQuant: Large Language Model Quantization via Exploiting End Loss Guidance*. 2025. arXiv: 2505.07004 [cs.LG]. URL: https://arxiv.org/abs/2505.07004.

[35] M. van Baalen, A. Kuzmin, M. Nagel, P. Couperus, C. Bastoul, E. Mahurin, T. Blankevoort, and P. Whatmough. "GPTVQ: The Blessing of Dimensionality for LLM Quantization." *arXiv preprint arXiv:2402.15319*, 2024.

[36] O. Ordentlich and Y. Polyanskiy. "Optimal Quantization for Matrix Multiplication." *arXiv preprint arXiv:2410.13780*, 2024.

[37] J. Kim, E. Ewer, T. Moon, J. Park, and D. Papailiopoulos. "Not All Bits Are Equal: Scale-Dependent Memory Optimization Strategies for Reasoning Models." *arXiv preprint arXiv:2510.10964*, 2025.

[38] Z. Allen-Zhu and Y. Li. "Physics of language models: Part 3.3, knowledge capacity scaling laws." *arXiv preprint arXiv:2404.05405*, 2024.

[39] J. X. Morris, C. Sitawarin, C. Guo, N. Kokhlikyan, G. E. Suh, A. M. Rush, K. Chaudhuri, and S. Mahloujifar. "How much do language models memorize?" *arXiv preprint arXiv:2505.24832*, 2025.

[40] D. Micciancio and O. Regev. "Worst-case to average-case reductions based on Gaussian measures." *SIAM journal on computing*, **37**(1), 2007, pp. 267–302.

[41] C. Ling, L. Luzzi, J.-C. Belfiore, and D. Stehlé. "Semantically secure lattice codes for the Gaussian wiretap channel." *IEEE Transactions on Information Theory*, **60**(10), 2014, pp. 6399–6416.

[42] A. Rényi. "On the dimension and entropy of probability distributions." *Acta Mathematica Academiae Scientiarum Hungarica*, **10**(1), 1959, pp. 193–215.