Entropy-Constrained Adaptive Stochastic Quantization

Ran Ben Basat[1,2] Yaniv Ben-Itzhak[2] Michael Mitzenmacher[3] Shay Vargaftik[2]

[1]University College London [2]VMware Research by Broadcom [3]Harvard University

Abstract

Adaptive stochastic quantization (ASQ) is a recently introduced quantization approach that optimizes the Mean Squared Error (MSE) for a given input while preserving unbiasedness. It is designed to alleviate the communication and memory bottlenecks of modern data and machine learning workloads, including model, gradient, and KV-cache compression and nearest-neighbor search. Further, practical systems can then compress quantized data with a lossless entropy encoder. However, existing unbiased methods, including ASQ, choose their quantization values without considering this later encoding stage, leaving accuracy on the table.

We formulate the Entropy Constrained Adaptive Stochastic Quantization (ECASQ) problem, which jointly selects adaptive quantization values to minimize MSE under an entropy budget and an unbiasedness constraint. We give an optimal dynamic program with $O(sd^2)$ time and $O(d^2)$ space for a length-d vector and at most s quantization values, and a GPU-friendly approximate dynamic program with $O(sd^2)$ time and $O(d)$ space. The approximation guarantees that the solution has an MSE no larger than the optimal solution that uses one fewer bit of entropy per entry. We also provide an iterative refinement procedure for the approximation solution that, in our experiments, yields near-optimal results while retaining a substantial speed advantage over our solver for the optimal solution.

1 Introduction

The training and deployment of machine learning (ML) models are often constrained by memory capacity, computational resources, and network bandwidth. To alleviate these constraints, quantization has become a cornerstone technique. By mapping high-precision values to low-width representations, quantization facilitates effective compression strategies across various ML workloads, including gradient compression [18, 23, 26, 32] and communication-efficient distributed training [11, 19, 24, 35], post-training quantization [15, 22], and KV-cache footprint reduction [25, 29], including disaggregated inference [34].

Three properties are especially desirable in this setting: (1) unbiasedness, (2) adaptivity to the input, and (3) optimization for the size of the entropy-encoded output. As we describe next, existing methods do not provide all three simultaneously.

Let $X = \langle x_1, \ldots, x_d \rangle \in \mathbb{R}^d$ be the input, and let $Q = \{q_1, \ldots, q_m\} \subset \mathbb{R}$, with $q_1 < \cdots < q_m$, be an ordered *codebook* of m quantization values. To enable unbiasedness, a valid Q must span X: $q_1 \leq \min_i x_i$ and $q_m \geq \max_i x_i$. Stochastic Quantization (SQ) rounds each x_i between its nearest values in Q, producing $\widehat{X} = \langle \widehat{x}_1, \ldots, \widehat{x}_d \rangle$ with $\mathbb{E}[\widehat{x}_i] = x_i$. Its distortion is $MSE(Q, X) = \mathbb{E}\left[\|X - \widehat{X}\|^2\right]$.

Coordinatewise unbiasedness, $\mathbb{E}[\widehat{X}] = X$, is desired for different applications, including for fast convergence under gradient compression [2, 10]. In Distributed Mean Estimation [30], it also makes errors cancel, allowing the MSE to decrease inversely with the number of participants [4, 31].[1]

[1]We note that there are other methods for achieving unbiased quantization, such as shared-randomness protocols [5], subtractive dithering [27] and Adaptive Unbiased Quantization [7]. None of these methods are optimized for entropy encoding.

Adaptive methods choose Q for the particular input X and improve substantially over predetermined codebooks [6, 14, 33]. Adaptive Stochastic Quantization (ASQ) minimizes $MSE(Q, X)$ over $|Q| \leq s$ given a bound s [33]; it is therefore optimal for fixed-length value identifiers, or when no entropy encoding follows. ASQ admits an optimal $O(d \cdot s)$-time algorithm [6].

Lossless entropy encoding, which is often follows quantization by practical systems, assigns shorter bit strings to more frequent values. For $q \in Q$, let $f_q = d^{-1} \sum_{i=1}^{d} \Pr[\widehat{x}_i = q]$. We define the average-entry entropy (in bits) as $H(\widehat{X}) = -\sum_{q \in Q} f_q \log_2 f_q$, with $0 \log_2 0 = 0$. Equivalently, this is the entropy of $\widehat{x}_I$ for a uniformly random coordinate I.

Entropy-aware adaptive methods include Entropy-Constrained Scalar and Vector Quantization [12, 17], broader vector-quantization frameworks [16], and neural compression [3]. These methods do not enforce unbiasedness.

A separate line of research in distributed optimization includes quantization schemes in which unbiased quantization is followed by entropy encoding. Notable examples include QSGD [2] and Distributed Mean Estimation with limited communication [30], but they are inherently *non-adaptive*, i.e., they rely on predetermined quantization grids (e.g., values scaled globally by a vector norm), which are generally less accurate and can result in excessive codebook sizes.

Randomized-transform approaches such as DRIVE and EDEN likewise apply scalar quantizers whose levels are prescribed up to scaling, rather than solving the input-specific ASQ codebook problem. Recent work further clarifies this line's relationship to TurboQuant and analyzes when fast randomized Hadamard transforms can replace uniform random rotations [8, 9, 31, 32].

This motivates the *Entropy Constrained Adaptive Stochastic Quantization* (ECASQ) problem. In addition to X, its inputs are a *permissible* alphabet $P \subset \mathbb{R}$ of size p (assumed to contain an encompassing codebook), a maximum codebook size $s \leq p$, and an average-entry entropy bit-budget $b \geq 0$. Specifically, ECASQ solves

$$\begin{aligned} \min_{Q \subseteq P} \quad & MSE(Q, X) \\ \text{subject to} \quad & |Q| \leq s,\ H(\widehat{X}) \leq b,\ \mathbb{E}[\widehat{X}] = X. \end{aligned} .$$

In addition to the entropy constraint, constraining the codebook size s is important to limit its representation overhead when compressing vectors of limited size. Also, working with smaller codebooks results in a faster ECASQ solution as well as quicker quantizing and dequantizing.

Solution Overview. The first step towards solving ECASQ is to introduce a Lagrange multiplier (as commonly done by previous entropy-encoding-aware works, e.g., [12, 17]), which allows us to consider, for some $\lambda \geq 0$, the relaxed problem, which we name λ-ECASQ.

$$\begin{aligned} \min_{Q \subseteq P} \quad & MSE(Q, X) + \lambda \cdot H(\widehat{X}) \\ \text{subject to} \quad & |Q| \leq s,\ \mathbb{E}[\widehat{X}] = X. \end{aligned} .$$

Intuitively, $\lambda = 0$ is the classic ASQ problem, while large λ values force the algorithm to prioritize low entropy over reducing error. Accordingly, one can vary λ to retrieve the convex hull of the Pareto-optimal MSE-Entropy trade-off, under the natural assumption that each entry is stochastically quantized to one of the encompassing quantization values.

We note that, generally, this Lagrangian optimization may not be able to find an optimal solution to the ECASQ problem for a given value b. Specifically, a respective λ value that results in the exact entropy constraint may not exist (but if some λ achieves a value b, that solution is optimal for that b). We expand on this issue in Section 7 and show that if 'time-sharing' (running two quantizers obtained from solving λ-ECASQ with two different λ values, on a suitable random partition of the input) is allowed, then we recover the ability to provide the optimal MSE for any entropy constraint b. In practice, however, the single Lagrangian solution is generally optimal or very close to optimal and avoids the need for two quantizers.

We first explain why the dynamic program (DP) used by optimal ASQ methods such as ZipML [33] and QUIVER [6] cannot be extended to consider entropy encoding. Instead, we present a novel DP that allows solving λ-ECASQ optimally. While a naive implementation of the new DP requires $O(p^3 \cdot s + d)$ time and $O(p^2 \cdot s + d)$ space, we show how to use existing DP acceleration and space-saving techniques such as SMAWK [1] and Hirschberg [20] to lower the asymptotics to $O(p^2 \cdot s + d)$ time and $O(p^2 + d)$ space.

To further reduce space and enable running on large p, we present an approximation algorithm which guarantees that it is at most 1 bit away from being optimal; that is, with an entropy bound of $b > 1$ bits, the approximate solution has an MSE that is not larger than that of an optimal λ-ecasq solution with an entropy bound of $b - 1$ bits and the same λ value. This new approximation algorithm requires $O(p^2 \cdot s + d)$ time and $O(p \cdot s + d)$ space, where the latter can be reduced to $O(p + d)$ using Hirschberg's method. While the asymptotic running time of this algorithm matches that of our optimal DP, it is orders of magnitude faster in practice due to being GPU-friendly and space efficient.

Finally, we show how the approximation algorithm's accuracy can be further improved by cascading the approximate optimization with a Lloyd-max-style iterative refinement phase. Through extensive evaluation over multiple input distributions, we demonstrate that the result is near-optimal after a handful of steps while remaining much faster and space-efficient than the optimal DP variant.

2 Preliminaries

Given an input $X \in \mathbb{R}^d$ and a set of quantization values $Q \subset R$, Stochastic Quantization (SQ) returns a quantized vector $\widehat{X} = \langle \widehat{x} \rangle_{x \in X}$ such that each $x \in X$ is unbiasedly quantized between its encompassing values in Q. In line with previous works (e.g., [6, 14, 33]), we hereafter assume that X is *sorted* (otherwise, we sort a copy of X and keep the original indices to quantize entries in the initial order). We further assume that the entries of X are distinct (otherwise, we can work with its weighted histogram). That is, denoting by $a_x = \max\{q \in Q \mid q \leq x\}$ and $b_x = \min\{q \in Q \mid q \geq x\}$ the relevant entries in Q, we have that $(\widehat{x} = a_x)$ if $(a_x = b_x)$ or $\widehat{x} = \begin{cases} b_x & \text{w.p. } (x - a_x)/(b_x - a_x) \\ a_x & \text{otherwise} \end{cases}$ if $(a_x \neq b_x)$. Since $\mathrm{Var}[\widehat{x}] = \Pr[\widehat{x} = b_x] \cdot (b_x - x)^2 + \Pr[\widehat{x} = a_x] \cdot (a_x - x)^2 = (b_x - x)(x - a_x)$, we have that the Mean Squared Error (MSE) of the quantization is

$$MSE(Q, X) = \sum_{x \in X} (b_x - x)(x - a_x).$$

An alternative error metric is the vector-Normalized MSE (vNMSE) [8, 9, 31], which is defined as

$$vNMSE(Q, X) = MSE(Q, X) / \|X\|^2 .$$

vNMSE is often a more convenient metric as it is scale free (i.e., $vNMSE(Q, X) = vNMSE(c \cdot Q, c \cdot X)$ for any $c \neq 0$) which allows comparing methods for various input dimensions.

Adaptive SQ (ASQ) is the problem where we are given just $X \in \mathbb{R}^d$ and $s \geq 2$ and are asked to output Q of size s that minimizes $MSE(Q, X)$ (or equivalently, $vNMSE(Q, X)$) for the specific X.

Optimal ASQ methods such as [6, 33] operate by solving the following dynamic program (DP); they define $dp_{ASQ}(i, j)$ as the minimal MSE attainable by stochastically quantizing the j smallest entries in X using i quantization values. They define $C[k, j] = \sum_{\ell=k}^{j} (x_j - x_\ell)(x_\ell - x_k)$ as the sum of variances of all entries in $[x_k, x_j]$ assuming that they are quantized between x_k and x_j. This formulation can then be solved using the recurrence for $dp_{ASQ}(i, j)$:

$$\begin{cases} \min\limits_{k \in \{1, \ldots, j\}} dp_{ASQ}(i-1, k) + C[k, j] & \text{If } i > 2 \\ C[k, j] & \text{Otherwise} \end{cases} . \tag{1}$$

Intuitively, this assumes that x_j is a quantization value and finds the optimal placement x_k of the next value. This dynamic program crucially relies on the fact that there exists an optimal solution Q^* in which $Q^* \subseteq X$, i.e., where all quantization values are entries in the input [6, 33]. This problem can be solved optimally in $O(s \cdot d)$ time and space [6]. For an integer n, we denote $[n] = \{1, \ldots, n\}$.

3 Entropy Constrained ASQ

To optimize the set of quantization values Q when an entropy encoding step will follow, we extend the ASQ problem as follows. The input for the Entropy Constrained Adaptive Stochastic Quantization (ECASQ) problem now includes (in addition to the parameter s that bounds $|Q|$) an entropy bound $b \in \mathbb{R}^+$ and a set of *permissible* quantization values $P \subset R$. We add the constraints that $H(\widehat{X}) \leq b$ and that $Q \subseteq P$. Here, $H(\widehat{X})$ is the entropy of the random vector of the quantized entries. Namely, for each $q \in Q$, let $f_q = \sum_{x \in X} \Pr[\widehat{x} = q]$ denote the expected frequency of q. Then the *average entry* entropy is defined as

$$H(\widehat{X}) = \sum_{q \in Q} -f_q/d \cdot \log_2 f_q/d.$$

For example, consider $X = \langle 0, 1, 10 \rangle$ and $Q = \{0, 10\}$; then 0 is surely quantized to 0 and 10 is quantized to 10, while the middle entry is quantized to 0 with probability $9/10$ and to 10 otherwise. The entropy is therefore $H(\widehat{X}) = -19/30 \log_2(19/30) - 11/30 \log_2(11/30) \approx 0.948$ bits per entry.

The introduction of P is necessary because, unlike in the ASQ case, the optimal solution may not be a subset of the input, and it also allows us to impose practical constraints on the encoding (e.g., by defining P as all values representable in BF16 or FP32). Notice that ECASQ generalizes ASQ, which is the special case $b = \log_2 s$ and $P = X$. Although one could remove the separate cardinality parameter by setting $s = |P|$, doing so ignores the size required for representing the codebook. The decoder must know both the selected dictionary Q and the entropy-code metadata; a cap on s therefore gives a direct, implementation-independent bound on this overhead. If a stored reconstruction value and its entropy-model descriptor use w and c bits, respectively, their metadata costs $(w+c)|Q|/d$ bits per entry. Since Q is ordered, a level's position implicitly supplies its symbol-to-value association, so no additional permutation is needed. Our experiments use $w = c = 16$.

Towards solving ECASQ, we follow previous entropy-aware quantization works (e.g., [12]) and introduce a Lagrangian multiplier $\lambda' > 0$ to write a revised cost function of $MSE(Q, X) + \lambda' \cdot H(\widehat{X})$. In practice, it is often comfortable to use $vNMSE(Q, X) + \lambda \cdot H(\widehat{X})$ as the cost, which is equivalent to the above when $\lambda = \lambda' / \|X\|^2$. Notice that this still generalizes ASQ (for $\lambda = 0$) and that by a binary search for λ we can respect the entropy constraint.

We note that the standard ASQ DP (1) is not suitable for the Lagrangian parameterization. This is because the new cost function is not separable, and the expected frequency (and thus, entropy) of a quantization value x_k depends on both adjacent quantization values, not just the one on its right (x_j). Accordingly, we design our optimal and approximate DPs to overcome this challenge.

In what follows, we derive optimal and approximate algorithms for ECASQ. In Section 4, we present an optimal algorithm for the problem that runs in $O(d^2 \cdot s)$ time and require $O(d^2)$ space. Subsequently, Section 5 features an approximation algorithm that requires just $O(d)$ space while producing a solution that, for $b > 1$, has an MSE which is upper bounded by the optimal solution with $b - 1$ bits. The generalization to $P \neq X$ appears in Appendix D.

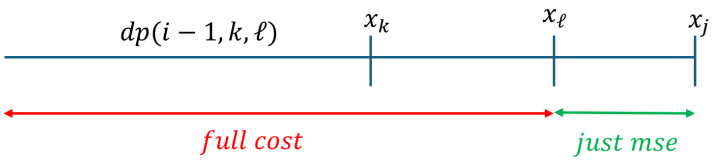


Figure 1: The structure of $dp_{OPT}(i,\ell,j)$: we look for the k that minimizes the cost of the prefix up to x_ℓ and only the MSE for $[x_\ell, x_j]$.

4 The Optimal Dynamic Program

The intuition behind our optimal DP is that, as expected frequency of a quantization value in the quantized vector $\widehat{X}$ only depends on the two adjacent values in Q, we can account for its contribution to the entropy if we track the last two quantization values.

For ease of presentation, we first assume that $P = X$, i.e., that the algorithm may only place quantization values on input entries, same as in the ASQ methods.

We present a new dynamic program that uses two helper arrays, for any $1 < a < b \in \{1, \ldots, d\}$:

$$U[1,b] = \sum_{i=1}^{b} \frac{x_i - x_1}{x_b - x_1} \quad , \quad U[a,b] = \sum_{i=a+1}^{b} \frac{x_i - x_a}{x_b - x_a}$$
$$D[1,b] = \sum_{i=1}^{b} \frac{x_b - x_i}{x_b - x_1} \quad , \quad D[a,b] = \sum_{i=a+1}^{b} \frac{x_b - x_i}{x_b - x_a} \; .$$

That is, $U[a,b]$ is the sum of all "round-up" probabilities (and $D[a,b]$, the round-down) of entries in the range, assuming two quantization values at x_a, x_b and none in between. The intuition is that the frequency of a value at x_ℓ, assuming that $x_k, x_\ell, x_j \in Q$ are consecutive, is $U(k,\ell) + D(\ell,j)$ in expectation. For all n, we set $U[n,n] = D[n,n] = 0$.

For the new dynamic program, we define $dp_{OPT}(i,\ell,j)$ to be the minimal cost for quantizing the ℓ smallest entries using $i-1$ quantization values that include x_ℓ, *plus* the MSE for quantizing $\{x_\ell, \ldots, x_j\}$ assuming that $x_\ell, x_j \in Q$ with no quantization values in between. This is illustrated in Figure 1.

For $y \in [0,1]$, let $h(y) = \begin{cases} -y \log_2 y & \text{if } \; y > 0 \\ 0 & \text{otherwise} \end{cases}$.

Assuming we solve $dp_{OPT}(i,\ell,j)$ for all $i \in [s], \ell, j \in [d]$, we have that the optimal cost is:

$$\min_{\ell \in [d]} \left(dp_{OPT}(i,\ell,d) + \lambda \times h\left(\frac{U(\ell,d)}{d}\right) \right).$$

$dp(i,\ell,j)$ is solved using the recurrence for $i \geq 3$:

$$dp_{OPT}(i,\ell,j) = C[\ell,j] + \min_{k \in [\ell]} T_{i,\ell,j}(k) \; ,$$

where

$$T_{i,\ell,j}(k) \triangleq dp_{OPT}(i-1,k,\ell) + \lambda \times h\left(\frac{U(k,\ell) + D(\ell,j)}{d}\right).$$

The stopping condition for $i = 3$ is

$$dp_{OPT}(3,\ell,j) = C[1,\ell] + C[\ell,j] + \lambda \left[h\left(\frac{D(1,\ell)}{d}\right) + h\left(\frac{U(1,\ell) + D(\ell,j)}{d}\right) \right].$$

Naively, computing the value of each (i, ℓ, j) entry requires $O(d)$ time (for checking all options for k), while there are $s \cdot d^2$ cells in the DP, giving runtime and space bounds of $O(d^3 \cdot s)$ and $O(d^2 \cdot s)$, respectively. Below, we improve these to $O(d^2 \cdot s)$ and $O(d^2)$ using general DP machinery. We note that the recursion considers duplicates in Q as $\ell \in [\ell]$; one can remove these to obtain a duplicate-free Q with $|Q| \leq s$.

4.1 Optimizing runtime using SMAWK

The SMAWK algorithm [1] is a general technique for accelerating DPs. The input for the algorithm is a matrix $M \in \mathbb{R}^{d \times d}$ that satisfies the quadrangle inequality (also known as Monge property):

$$\forall \mathtt{a} \leq \mathtt{b} \leq \mathtt{c} \leq \mathtt{d} : M_{\mathtt{a},\mathtt{c}} + M_{\mathtt{b},\mathtt{d}} \leq M_{\mathtt{a},\mathtt{d}} + M_{\mathtt{b},\mathtt{c}}.$$

The algorithm then finds the row minima in $O(d)$ time. That is, it outputs $k_1, \ldots, k_d$ such that k_y is the smallest value in the y'th row.

An important property of the quadrangle inequality is stated in the following lemma (proved in Appendix A).

Lemma 1. *Let g be a concave function and let v, m be monotonically decreasing functions then $G(k, j) \triangleq g(v(k) + m(j))$ satisfies the quadrangle inequality.*

Fix i and ℓ. Let the admissible predecessor indices be $K_\ell = \{k : 1 \leq k < \ell\}$ and the admissible right endpoints be $J_\ell = \{j : \ell < j \leq d\}$. Denote

$$A(k) = dp_{\mathrm{OPT}}(i-1, k, \ell)$$

and

$$G(k, j) = \lambda \cdot h\left(\frac{U(k, \ell) + D(\ell, j)}{d}\right).$$

For our DP, define

$$M_{k,j} \triangleq T_{i,\ell,j}(k) = A(k) + G(k, j).$$

The matrix M is not necessarily Monge in the natural order of the columns, since $D(\ell, j)$ is monotone increasing in j. We thus apply SMAWK on reversed column order. Let

$$\rho_\ell(r) = d + \ell + 1 - r, \qquad r \in J_\ell,$$

be the order-reversing bijection of J_ℓ, and define $\widetilde{M}_{k,r} = M_{k,\rho_\ell(r)}$. The following is proved in Appendix B.

Lemma 2. *For fixed i and ℓ, the column-reversed matrix $\widetilde{M}$ satisfies the quadrangle inequality.*

Since $\widetilde{M}$ is Monge, so is its transpose. Therefore SMAWK can be run on $\widetilde{M}^{\mathsf{T}}$ to compute, in $O(|K_\ell| + |J_\ell|) = O(d)$ time, a minimizing predecessor

$$k_j^\ell \in \arg\min_{k<\ell} M_{k,j}$$

for every $j \in J_\ell$. We then fill the DP entries as

$$dp_{\mathrm{OPT}}(i, \ell, j) = C[\ell, j] + T_{i,\ell,j}(k_j^\ell).$$

For $i > 3$, we invoke SMAWK for every $\ell \in [d]$ and use its output $\left\{k_j^\ell\right\}$ to fill in d entries in the matrix, as:

$$dp_{OPT}(i, \ell, j) = C[\ell, j] + T(k_j^\ell).$$

Overall, as we invoke the $O(d)$-time SMAWK algorithm for each $i \in [s], \ell \in [d]$, the runtime becomes $O(d^2 \cdot s)$.

4.2 Optimizing space using Hirschberg

We now explain how to apply Hirschberg's algorithm [1975] to our optimal DP.

The first step is to observe that while we defined $dp_{OPT}(i, \ell, j)$ to be the minimal cost for quantizing the ℓ smallest entries in X (henceforth referred to as the "forward algorithm"), an equivalent algorithm can define $dp^B_{OPT}(i, \ell, j)$ to be the cost for the largest entries (the "backward algorithm"). Specifically, we define $dp^B_{OPT}(i, \ell, j)$ as the minimal cost for quantizing the $d - j$ *largest* entries using $i - 1$ quantization values that include x_j, *plus* the MSE for quantizing $\{x_\ell, \ldots, x_j\}$ assuming that $x_\ell, x_j \in Q$ with no quantization values in between. The recurrence is then, for $i \geq 3$:

$$dp^B_{OPT}(i, \ell, j) = C[\ell, j] + \min_{k \in [d] \setminus [j-1]} T^B_{i,\ell,j}(k) \ ,$$

where

$$T^B_{i,\ell,j}(k) \triangleq dp^B_{OPT}(i-1, j, k) + \lambda \times h\left(\frac{U(\ell, j) + D(j, k)}{d}\right).$$

The Hirschberg method follows a divide-and-conquer approach: it uses the forward algorithm to compute $dp_{OPT}(i, \ell, j)$, for all $\ell, j \in [d]$ and $i \in [\lfloor s/2 \rfloor]$. It then uses the backward algorithm to compute $dp^B_{OPT}(i, \ell, j)$ for all $i, \ell, j \in [d], \in [\lceil s/2 \rceil]$. Importantly, the algorithm only keeps a pair of layers $(i, i+1)$ as it progresses, keeping the memory bounded by $O(d^2)$.

Next, denoting $t = \lfloor s/2 \rfloor$, it finds the values ℓ^*, j^* that minimize

$$dp_{OPT}(t+1, \ell^*, j^*) + dp^B_{OPT}(s - t + 1, \ell^*, j^*) - C[\ell^*, j^*] \ .$$

Intuitively, the optimal minimum is obtained when ℓ^*, j^* are the middle quantization values in the optimal solution.

$dp_{OPT}(\lfloor s/2 \rfloor, \ell^*, j^*)$ and $dp^B_{OPT}(\lceil s/2 \rceil, \ell^*, j^*)$ are then recursively computed while reusing memory across the nested calls (i.e., we keep just ℓ^*, j^* from the initial run and once $dp_{OPT}(\lfloor s/2 \rfloor, \ell^*, j^*)$ terminates, we reuse the memory except the inner minimizer indices.)

We state the resulting asymptotics (proof in Appendix C).

Lemma 3. *Using Hirschberg's algorithm, the space complexity of our DP is $O(d^2)$ while the time complexity remains $O(d^2 \cdot s)$.*

General permissible sets. The optimal DP extends from $P = X$ to any sorted permissible set $P = \langle p_1, \ldots, p_p \rangle$ by indexing states by permissible values and adjusting the boundary conditions (see Appendix D for the full construction). The resulting optimal algorithm runs in $O(p^2 \cdot s + d)$ time and uses $O(p^2 + d)$ space.

5 A Space-Efficient Approximation Algorithm

The optimal DP from Section 4 keeps enough state to charge the entropy contribution of each quantization value only after both of its neighboring quantization values are known. We now present a more space-efficient alternative. The main idea is to optimize a slightly stronger entropy surrogate whose contribution is local to each interval between consecutive quantization values. This removes the need to keep two adjacent quantization values in the state.

Let $Q = \{q_1 < \ldots < q_t\}$ be a feasible set of quantization values, with $q_1 \leq x_1$, $q_t \geq x_d$, and $t \leq s$. For each coordinate x, define its interval index $I_x \in \{1, \ldots, t-1\}$ such that $q_{I_x} < x \leq q_{I_x+1}$.[2] Let $B_x \in \{0, 1\}$

[2] If $x = q_1$, we define $I_x = 1$.

indicate the stochastic rounding decision, where $B_x = 1$ if x is rounded up to q_{I_x+1} and $B_x = 0$ otherwise. Thus $\widehat{x} = q_{I_x+B_x}$.

The approximation comes from optimizing the joint entropy $H(I, B)$ rather than $H(\widehat{X})$, where the randomness is over a uniformly chosen coordinate and the stochastic rounding. For a fixed Q, the map $(I, B) \mapsto (\widehat{X}, B)$ is a bijection and therefore

$$H(I, B) = H(\widehat{X}, B) = H(\widehat{X}) + H(B \mid \widehat{X}).$$

Since $B = \{B_x\}$ has one bit per entry,

$$H(\widehat{X}) \le H(I, B) \le H(\widehat{X}) + 1. \tag{2}$$

Before delving into the algorithm, there are several important things to note here: First, finding the minimal MSE solution under the constraint $H(I, B) \le b$ immediately yields an MSE better than the optimal solution with one fewer bit per entry (i.e., for $H(\widehat{X}) \le b - 1$). This is because, by Equation (2), the optimal solution for $H(\widehat{X}) \le b-1$ is a feasible solution for $H(I, B) \le b$; if our algorithm yields a different solution, its MSE cannot be larger.

Second, while we are optimizing $H(I, B)$, this is done to find a good set Q, but the quantizer still tries to save a bit (pun intended) by compressing the quantized values in $\widehat{X}$; thus, our actual overhead is $H(B \mid \widehat{X})$.

Third, while this overhead is bounded by one bit per entry, its actual value is dependent on X and Q. On one extreme, consider the case where X has $0, 1$ and $d/2-2$ occurrences of $(1/2-\epsilon)$ and $(1/2+\epsilon)$ each. If $Q = \{0, 1/2, 1\}$, then all the $(1/2 \pm \epsilon)$ are rounded to $1/2$, giving $H(\widehat{X}) \approx 0$. In contrast, $H(B \mid \widehat{X}) \approx 1$, since knowing that $\widehat{x} = 1/2$ reveals little about whether $x = 1/2 - \epsilon$ or $x = 1/2 + \epsilon$. For the other extreme, let X have $0, 1$ and $d/2 - 2$ occurrences of (ϵ) and $(1 - \epsilon)$ each. If $Q = \{0, 1/2, 1\}$, then all the (ϵ)s are rounded to 0 and the $(1 - \epsilon)$ entries are rounded to 1, so $H(B \mid \widehat{X}) \approx 0$ since $\widehat{x}$ implies whether $B_x = 0$ or $B_x = 1$. In the evaluation, we quantify the magnitude of $H(B \mid \widehat{X})$ in practice for input sizes in which running the optimal algorithm is feasible.

Let $P = \langle p_1, \ldots, p_p \rangle$ be sorted and denote, for $a < b$, $X_{a,b} = \{x \in X \mid p_a < x \le p_b\}$. For $b > 1$, we define $X_{1,b} = \{x \in X \mid p_1 \le x \le p_b\}$, and for $1 < a < b$, define $X_{a,b} = \{x \in X \mid p_a < x \le p_b\}$. We use this half-open convention only to avoid double-counting entries that are themselves permissible quantization values. The MSE contribution is unaffected by the choice of endpoint convention, since entries equal to a quantization value have zero variance. We define

$$C_P[a, b] = \sum_{x \in X_{a,b}} (p_b - x)(x - p_a),$$

and similarly

$$U_P[a, b] = \sum_{x \in X_{a,b}} \frac{x - p_a}{p_b - p_a}, \qquad D_P[a, b] = \sum_{x \in X_{a,b}} \frac{p_b - x}{p_b - p_a}.$$

Thus, if p_k, p_ℓ, p_j are three consecutive quantization values, the expected frequency of p_ℓ is $U_P[k, \ell] + D_P[\ell, j]$.

For a non-first interval $(p_a, p_b]$, the two interval-bit symbols have probabilities $U_P[a, b]/d$ and $D_P[a, b]/d$, so define

$$\Phi_P[a, b] = h\left(\frac{U_P[a, b]}{d}\right) + h\left(\frac{D_P[a, b]}{d}\right).$$

For the first interval, entries equal to the first quantization value are assigned to the down symbol, and we use

$$\Phi_P^{\text{first}}[a, b] = h\left(\frac{U_P[a, b]}{d}\right) + h\left(\frac{D_P[a, b] + n_a}{d}\right),$$

where $n_a = |\{x \in X \mid x = p_a\}|$. Consequently, if $Q = \{p_{a_1} < \ldots < p_{a_t}\}$, then

$$H(I, B) = \Phi_P^{\text{first}}[a_1, a_2] + \sum_{r=2}^{t-1} \Phi_P[a_r, a_{r+1}]. \tag{3}$$

This is the separability property that the true entropy $H(\widehat{X})$ does not have.

We now write the Lagrangian DP for a fixed multiplier $\lambda \geq 0$. Let $L = \{a \in [p] \mid p_a \leq x_1\}$ and $R = \{a \in [p] \mid p_a \geq x_d\}$. Define $dp_{APX}(i, j)$ to be the minimum surrogate cost of all partial solutions that use i quantization values and whose rightmost quantization value is p_j. The initialization for two quantization values is

$$dp_{APX}(2, j) = \min_{\substack{a \in L \\ a < j}} \left\{ C_P[a, j] + \lambda \, \Phi_P^{\text{first}}[a, j] \right\}. \tag{4}$$

For $i \geq 3$, the recurrence is

$$dp_{APX}(i, j) = \min_{k < j} \Big\{ dp_{APX}(i-1, k) + C_P[k, j] + \lambda \, \Phi_P[k, j] \Big\}. \tag{5}$$

The best value using at most s quantization values is then

$$\min_{2 \leq i \leq s} \min_{j \in R} dp_{APX}(i, j). \tag{6}$$

If exactly s values are required, the outer minimum is replaced by $i = s$.

The recurrence has the same structure as the standard ASQ recurrence: once the previous quantization value p_k and the new value p_j are fixed, the cost of the new interval is completely determined. In Appendix D, using preprocessing of $O(d + p)$ time and space, we show how each of $C_P[k, j]$, $U_P[k, j]$, and $D_P[k, j]$ is computed in $O(1)$ time. Thus, one DP layer costs $O(p^2)$ time, and all s layers cost $O(sp^2 + d)$ time. We store only the previous and current DP layers, as well as the $O(p + d)$ prefix information, giving $O(p + d)$ space for the value computation. In particular, when $P = X$, this is $O(sd^2)$ time and $O(d)$ space.

To recover the quantization values while preserving the same asymptotic space bound, we use the Hirschberg-based traceback idea as in Section 4, specialized to the one-dimensional recurrence in (5). A forward pass computes the best cost up to the middle number of quantization values, a backward pass computes the corresponding suffix costs, and their sum identifies a middle quantization value. Recursing on the two sides reconstructs Q while reusing the same two DP layers. This keeps the peak memory at $O(p + d)$ and the running time at $O(sp^2 + d)$. Note that while the runtime is asymptotically equal to that of the optimal algorithm of Section 4, the quadratic space reduction is critical if p is large.

For a target entropy budget b, we tune λ and keep the lowest-MSE solution whose surrogate entropy $H(I, B)$ is at most b. The next lemma summarizes the resulting guarantee.

Lemma 4. *Let $OPT(\beta)$ be the minimum MSE among feasible quantizers with $H(\widehat{X}) \leq \beta$, and, for $b > 1$, let $Q_{APX}(b)$ be the minimum-MSE quantizer with $H(I, B) \leq b$. Then $H(\widehat{X}_{Q_{APX}(b)}) \leq b$ and*

$$MSE(Q_{APX}(b), X) \leq OPT(b - 1),$$

where $OPT(\beta) = \infty$ if the budget β is infeasible.

Proof. The feasibility of the returned quantizer for the original entropy budget follows directly from (2): $H(\widehat{X}_{Q_{APX}(b)}) \leq H(I, B) \leq b$.

Let Q^- be an optimal solution for budget $b-1$. Then $H(\widehat{X}_{Q^-}) \leq b-1$. By (2), the same quantizer satisfies $H(I, B) \leq H(\widehat{X}_{Q^-}) + 1 \leq b$, so it is feasible for the surrogate problem with budget b. Since $Q_{APX}(b)$ minimizes MSE over the surrogate-feasible quantizers, its MSE is at most $MSE(Q^-, X) = OPT(b - 1)$. □

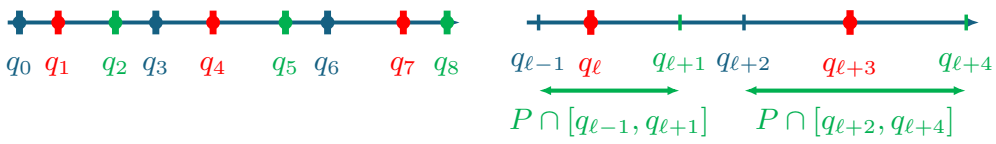


Figure 2: Three-color refinement of a quantizer. (a) The quantization values in Q are partitioned by index into Q_0 (blue), Q_1 (red), and Q_2 (green). (b) During the Q_1 step, each red value is optimized over its range while neighboring values remain fixed. Consecutive red values have disjoint ranges, so all values in Q_1 are updated in parallel; the other two substeps are analogous.

6 The Refinement Algorithm

Due to its space efficiency and GPU-friendliness, the approximation algorithm described above allows running ECASQ on large inputs and data types. To alleviate the accuracy gap between its output and the optimal algorithm of Section 4 while maintaining the speed advantage, we iteratively improve the cost while maintaining unbiasedness.

Our starting point is a feasible solution Q (e.g., one that is generated by the approximation algorithm). [3] At each iteration, Q is updated by improving one of three sets: $Q_0 = \{q_{3\cdot k} \mid k \in \mathbb{N}, 3k < s\}$, $Q_1 = \{q_{3\cdot k+1} \mid k \in \mathbb{N}, 3k+1 < s\}$, $Q_2 = \{q_{3\cdot k+2} \mid k \in \mathbb{N}, 3k+2 < s\}$. When changing the position of q_ℓ, we use binary search to improve its position in $P \cap [q_{\ell-1}, q_{\ell+1}]$, assuming that $q_{\ell-1}$ and $q_{\ell+1}$ remain fixed. This is achieved by minimizing the sum of variances of all entries $x \in X \cap [q_{\ell-1}, q_{\ell+1}]$ plus λ times the expected normalized frequency of q_ℓ, which can be efficiently computed using U_P, D_P (Appendix D).

We choose these three sets as the quantization values in Q_i are independent from each other: moving one does not change the expected frequency of another, and thus we can optimize the values of a Q_i in parallel (while keeping other values fixed) on the GPU, as illustrated in Figure 2.

Namely, let q_{3k+i} and $q_{3(k+1)+i}$ be consecutive values in Q_i. Then small entries $(X \cap [x_1, q_{3k+i+1}])$ cannot be quantized to $q_{3(k+1)+i}$, while large entries $(X \cap [q_{3k+i+2}, x_d])$ cannot be quantized to q_{3k+i}, yielding a complete separation of the MSE and entropy across Q_i.

While the refinement algorithm is not guaranteed to reach the optimal solution, its output is always at least as good as the starting Q, and as we show in the evaluation (Section 8), it usually is near-optimal while being very quick compared to the approximate solution itself.

7 Optimality with Two-Way Time-Sharing

In this section, we explain the limitations of using the Lagrange relaxation, but also explain how it yields an optimal solution *for any* b when time-sharing is allowed.

From Lagrange Multipliers to Entropy Constraints. By Everett's theorem [13], if Q_λ globally minimizes $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$ and happens to satisfy $H(\widehat{X}) = b$, then Q_λ is globally optimal for the hard-constrained problem with entropy budget b. Thus, whenever the multiplier search hits the target entropy exactly, no time-sharing is needed. More generally, optimizing $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$ over λ recovers the supported points of the entropy-vNMSE frontier, but it need not return the optimal single quantizer for every hard entropy budget b (we provide a counterexample in Appendix E.1). Time-sharing fills the entropy levels skipped by the λ search: it convexifies the feasible rate-distortion region, and every boundary point

[3] Without loss of generality we assume that $|Q| = s$; otherwise one can duplicate one of the entries in Q without quantizing to it.

is attained by a mixture of at most two supported quantizers. Hence a sweep over λ recovers an optimal two-way time-share for every entropy constraint.

Quantizing to non-adjacent values in Q. Our algorithms use ordinary stochastic quantization between the two adjacent values of Q that encompass each input. For a single quantizer, allowing arbitrary unbiased rounding to non-adjacent values can yield strictly lower vNMSE under the same entropy constraint (we give an example in Appendix E.2).

Quantizing to non-adjacent values under two-way time-sharing. To regain optimality for non-supported values of b in the strict entropy bound formulation while still solving the λ-ECASQ problem and rounding to adjacent values in Q, we leverage *time-sharing* [12]. In our application, time-sharing means running two quantizers on a random partition of the input (determined by a shared pseudorandom number generator seed). A standard observation is that time-sharing recovers the optimal entropy-MSE tradeoff. Formally, for an entropy bound b, let Q^- be the best quantizer with entropy $b^- \leq b$ that can be derived from λ-ECASQ for some λ, and similarly let Q^+ be the quantizer attainable for some λ' with the lowest rate $b^+ \geq b$. Then if for each entry we independently use Q^+ with probability $\frac{b-b^-}{b^+-b^-}$, the MSE is guaranteed to be no larger than any single quantizer Q^* with entropy b, because the vNMSE is linear in the above probability. We prove a more general property that is specific to our problem: under time-sharing, it is enough to consider pairs of quantizers that only quantize to adjacent values in their quantization value sets (proof is in Appendix E.3).

Proposition 5. *Let $Q_1, Q_2 \subseteq P$ satisfy $|Q_1|, |Q_2| \leq s$, and let K_1, K_2 be arbitrary unbiased rounding rules into Q_1, Q_2. For every $\alpha \in [0,1]$, there exist two sets $S_1, S_2 \subseteq P$, each of size at most s, and $\beta \in [0,1]$, such that ordinary adjacent stochastic quantization with S_1, S_2 satisfies*

$$(1)\ \beta R(S_1) + (1-\beta)R(S_2) \leq \alpha R(K_1) + (1-\alpha)R(K_2),$$

$$(2)\ \beta D(S_1) + (1-\beta)D(S_2) \leq \alpha D(K_1) + (1-\alpha)D(K_2).$$

Here, $R(S)$ and $D(S)$ denote the entropy and vNMSE of ordinary adjacent stochastic quantization with S.

We provide an evaluation of the gain attainable by time-sharing in Appendix F.7.

8 Evaluation

Setup. Experiments ran on a 12-core Apple M2 with 32 GB RAM under macOS 15.7.2 (arm64); CPU benchmarks used eight threads with Python 3.9.6 and PyTorch 2.8.0.

We evaluate BF16 synthetic vectors of size $d = 16{,}384$ over five seeds and four distributions, together with 100 real-model tensor samples from five model families. We compare Optimal ECASQ, Approx ECASQ, five-round refined Approx ECASQ, and entropy-coded QUIVER, QSGD, and uniform stochastic quantization. We report the ideal total rate $R_{\text{tot}} = H(\widehat{X}) + 32|Q|/d$ bits/value, which includes a 16-bit BF16 reconstruction value and a 16-bit entropy-model descriptor per active level. Additional details and results appear in Appendix F.

Matched-rate vNMSE. Across the 12 distribution-rate comparisons in Figure 3, refinement reduces Approx ECASQ's mean excess vNMSE from 4.97% to 1.37% on the five-seed aggregate curves, with a 4.06% maximum. Unrefined Approx also outperforms every non-ECASQ baseline in mean at every rate: its per-rate means are 4.82–5.05%, versus 23.23–43.36% for the strongest baseline. Pointwise, it beats QSGD and QUIVER in all 12 comparisons and uniform quantization in 11 of 12.

Runtime. At $d = 262{,}144$, Figure 4 shows that Optimal ECASQ, Approx, and refined Approx take 6.39, 0.26, and 0.33 seconds on average: $24\times$ and $19\times$ speedups. The gains hold for every tested distribution, ranging from 13-$34\times$ for Approx and 8-$28\times$ after refinement. Other settings are evaluated in Appendix F.

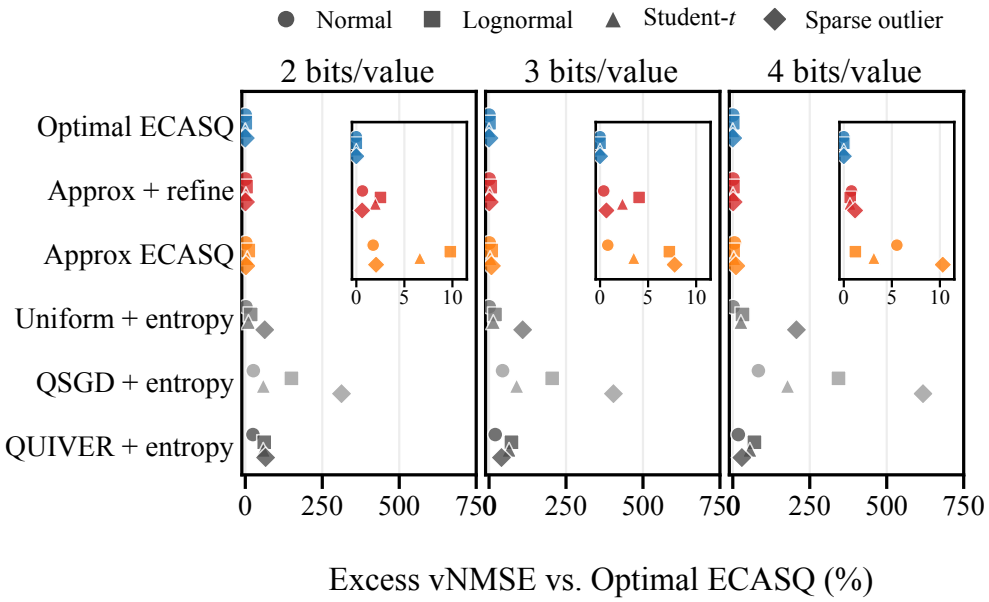


Figure 3: Matched-rate excess vNMSE (five seeds) relative to Optimal ECASQ on synthetic BF16 vectors ($d = 16{,}384$, at most $s = 64$ quantization values). Lower is better.

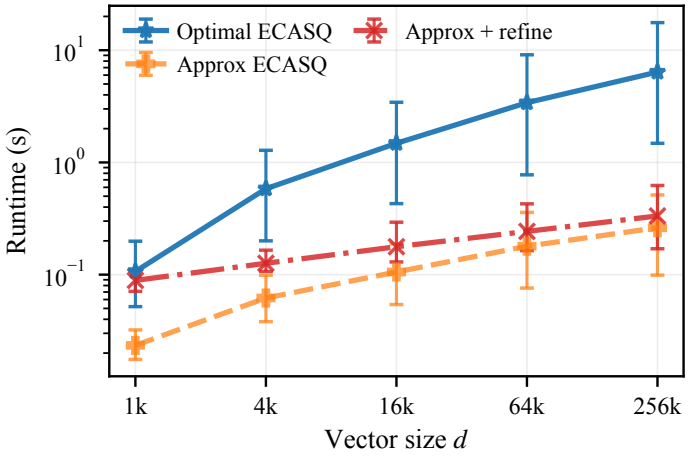


Figure 4: Runtime for BF16 inputs, $s = 64$, and $\lambda = 0.1$. Marks average the four distribution-specific means (five seeds each), and error bars span their min-max range.

Real models. We also quantize weight, activation, and KV-cache tensors from Qwen2.5-0.5B/1.5B, Gemma-3-1B-IT, Llama-3.2-1B, and Phi-3.5-Mini, improving througout all tasks. Additional details and results appear in Appendix F.

AI usage disclaimer. We have used ChatGPT and Codex to polish the writing of this paper and help with proof verification and simplification as well as evaluation design and execution.

References

[1] Alok Aggarwal, Maria Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix searching algorithm. In *Proceedings of the second annual symposium on Computational geometry*, pages 285–292, 1986.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *Advances in Neural Information Processing Systems*, 30:1709–1720, 2017.

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end Optimized Image Compression. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01704.

[4] Ran Ben Basat, Shay Vargaftik, Amit Portnoy, Gil Einziger, Yaniv Ben-Itzhak, and Michael Mitzenmacher. Accelerating Federated Learning with Quick Distributed Mean Estimation. In *International Conference on Machine Learning*, 2024.

[5] Ran Ben Basat, Michael Mitzenmacher, and Shay Vargaftik. How to Send a Real Number Using a Single Bit (And Some Shared Randomness). In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198, pages 25:1–25:20, 2021.

[6] Ran Ben-Basat, Yaniv Ben-Itzhak, Michael Mitzenmacher, and Shay Vargaftik. Optimal and Approximate Adaptive Stochastic Quantization. *Advances in Neural Information Processing Systems*, 37: 94265–94291, 2024.

[7] Ran Ben Basat, Yaniv Ben-Itzhak, Michael Mitzenmacher, and Shay Vargaftik. Better than Optimal: Improving Adaptive Stochastic Quantization Using Shared Randomness. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(3):49:1–49:44, 2025. doi: 10.1145/3771564. URL https://doi.org/10.1145/3771564.

[8] Ran Ben Basat, Yaniv Ben-Itzhak, Gal Mendelson, Michael Mitzenmacher, Amit Portnoy, and Shay Vargaftik. A Note on TurboQuant and the Earlier DRIVE/EDEN Line of Work. *arXiv preprint arXiv:2604.18555*, 2026. URL https://arxiv.org/abs/2604.18555.

[9] Ran Ben Basat, William Kuszmaul, Michael Mitzenmacher, Amit Portnoy, and Shay Vargaftik. Quantizing With Randomized Hadamard Transforms: Efficient Heuristic Now Proven. *arXiv preprint arXiv:2605.06014*, 2026. URL https://arxiv.org/abs/2605.06014.

[10] Roberto L Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native FP4 Training Can Be Optimal for Large Language Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[11] Xiaoqi Chen, Shay Vargaftik, and Ran Ben Basat. When ML Training Cuts Through Congestion: Just-in-Time Gradient Compression via Packet Trimming. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 177–185, 2024.

[12] Philip Chou, Tom Lookabaugh, and Robert Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–42, 1989. URL https://ieeexplore.ieee.org/document/15326.

[13] Hugh Everett, III. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11(3):399–417, 1963. doi: 10.1287/opre.11.3.399.

[14] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive Gradient Quantization for Data-parallel SGD. *Advances in neural information processing systems*, 33:3174–3185, 2020.

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

[16] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 1992. URL https://link.springer.com/book/10.1007/978-1-4615-3626-0.

[17] Herbert Gish and John N. Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, 1968. URL https://ieeexplore.ieee.org/document/1054193.

[18] Wenchen Han, Shay Vargaftik, Michael Mitzenmacher, Brad Karp, and Ran Ben Basat. Beyond Throughput and Compression Ratios: Towards High End-to-end Utility of Gradient Compression. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 186–194, 2024.

[19] Wenchen Han, Shay Vargaftik, Michael Mitzenmacher, and Ran Ben Basat. DynamiQ: Accelerating Gradient Synchronization Using Compressed Multi-hop All-reduce. In *Proceedings of the ACM SIGCOMM 2026 Conference*, 2026. doi: 10.1145/3789240.3829148. URL https://doi.org/10.1145/3789240.3829148.

[20] Daniel S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[21] Nikita Ivkin, Ran Ben Basat, Zaoxing Liu, Gil Einziger, Roy Friedman, and Vladimir Braverman. I know what you did last summer: Network monitoring using interval queries. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–28, 2019.

[22] Yongkweon Jeon, Chungman Lee, Kyungphil Park, and Ho-young Kim. A Frustratingly Easy Post-Training Quantization Scheme for LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14446–14461, 2023.

[23] Jakub Konečnỳ and Peter Richtárik. Randomized Distributed Mean Estimation: Accuracy vs. Communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018.

[24] Minghao Li, Ran Ben Basat, Shay Vargaftik, ChonLam Lao, Kevin Xu, Michael Mitzenmacher, and Minlan Yu. THC: Accelerating Distributed Deep Learning Using Tensor Homomorphic Compression. In *USENIX Symposium on Networked Systems Design and Implementation*, 2024.

[25] Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. In *NeurIPS*, 2024.

[26] Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M Roy. NUQSGD: Provably Communication-efficient Data-parallel SGD via Nonuniform Quantization. *The Journal of Machine Learning Research*, 22(114):1–43, 2021.

[27] Joshua Rapp, Robin M. A. Dawson, and Vivek K. Goyal. Estimation From Quantized Gaussian Measurements: When and How to Use Dither. *IEEE Transactions on Signal Processing*, 67(13):3424–3438, 2019. doi: 10.1109/TSP.2019.2916046. URL https://doi.org/10.1109/TSP.2019.2916046.

[28] Rana Shahout, Roy Friedman, and Ran Ben Basat. Together is better: Heavy hitters quantile estimation. *Proceedings of the ACM on Management of Data*, 1(1):1–25, 2023.

[29] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-Throughput Generative Inference of Large Language Models With a Single GPU. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[30] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed Mean Estimation With Limited Communication. In *International Conference on Machine Learning*, volume 70, pages 3329–3337. PMLR, 2017.

[31] Shay Vargaftik, Ran Ben-Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben-Itzhak, and Michael Mitzenmacher. Drive: One-bit Distributed Mean Estimation. *Advances in Neural Information Processing Systems*, 34:362–377, 2021.

[32] Shay Vargaftik, Ran Ben Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben Itzhak, and Michael Mitzenmacher. EDEN: Communication-Efficient and Robust Distributed Mean Estimation for Federated Learning. In *International Conference on Machine Learning*, pages 21984–22014. PMLR, 2022.

[33] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. ZipML: Training Linear Models with End-to-End Low Precision, and a Little Bit of Deep Learning. In *Proceedings of the 34th International Conference on Machine Learning*, ICML 2017, pages 4035–4043, 2017. URL https://proceedings.mlr.press/v70/zhang17e.html.

[34] Zeyu Zhang, Haiying Shen, Shay Vargaftik, Ran Ben Basat, Michael Mitzenmacher, and Minlan Yu. HACK: Homomorphic Acceleration via Compression of the Key-Value Cache for Disaggregated LLM Inference. *arXiv preprint arXiv:2502.03589*, 2025. URL https://arxiv.org/abs/2502.03589.

[35] Yutong Zhao, Noga H. Rotman, Gianni Antichi, and Ran Ben Basat. ML-for-ML. *arXiv preprint arXiv:2608.06046*, 2026. URL https://arxiv.org/abs/2608.06046.

A Proof of Lemma 1

We restate the lemma for convenience.

Lemma 1. *Let g be a concave function and let v, m be monotonically decreasing functions then $G(k,j) \triangleq g(v(k)+m(j))$ satisfies the quadrangle inequality.*

Proof. We show that for every $k < k'$ and $j < j'$,

$$G(k,j) + G(k',j') \leq G(k,j') + G(k',j).$$

Since v and m are monotonically decreasing,

$$v(k) \geq v(k') \qquad \text{and} \qquad m(j) \geq m(j').$$

Define

$$\Delta_v \triangleq v(k) - v(k') \geq 0, \qquad \Delta_m \triangleq m(j) - m(j') \geq 0.$$

Also define

$$A \triangleq v(k) + m(j), \qquad B \triangleq v(k) + m(j'),$$
$$C \triangleq v(k') + m(j), \qquad D \triangleq v(k') + m(j').$$

Then

$$A = D + \Delta_v + \Delta_m, \qquad B = D + \Delta_v, \qquad C = D + \Delta_m,$$

and therefore

$$A + D = B + C.$$

If $\Delta_v + \Delta_m = 0$, then $A = B = C = D$, and the claim is immediate. Assume now that $\Delta_v + \Delta_m > 0$, and let

$$\theta \triangleq \frac{\Delta_m}{\Delta_v + \Delta_m} \in [0, 1].$$

A direct calculation shows that

$$B = \theta D + (1 - \theta)A, \qquad C = (1 - \theta)D + \theta A.$$

Since g is concave, we have

$$g(B) \geq \theta g(D) + (1 - \theta)g(A),$$

and

$$g(C) \geq (1 - \theta)g(D) + \theta g(A).$$

Adding these two inequalities yields

$$g(B) + g(C) \geq g(A) + g(D).$$

Substituting the definitions of A, B, C, D, we obtain

$$g\big(v(k) + m(j')\big) + g\big(v(k') + m(j)\big) \geq g\big(v(k) + m(j)\big) + g\big(v(k') + m(j')\big).$$

By the definition of G, this is exactly

$$G(k, j') + G(k', j) \geq G(k, j) + G(k', j'),$$

or equivalently,

$$G(k, j) + G(k', j') \leq G(k, j') + G(k', j).$$

Hence G satisfies the quadrangle inequality. □

B Proof of Lemma 2

We restate the lemma for convenience.

Lemma 2. *For fixed i and ℓ, the column-reversed matrix $\widetilde{M}$ satisfies the quadrangle inequality.*

Proof. It suffices to prove the claim for the entropy term, since adding the row-dependent term $A(k)$ preserves the quadrangle inequality.

Define

$$v(k) \triangleq U(k,\ell), \qquad \widetilde{m}(r) \triangleq D(\ell, \rho_\ell(r)),$$

and

$$g(z) \triangleq \lambda h(z/d).$$

Because $h(0) = 0$ and $h(y) = -y \log_2 y$ for $y > 0$, the function h is concave on $[0,1]$. Moreover, every argument used below satisfies

$$0 \le U(k,\ell) + D(\ell, \rho_\ell(r)) \le d.$$

Therefore, $g(z) = \lambda h(z/d)$ is concave on the entire range of arguments appearing in the matrix.

We first show that $v(k)$ is monotone decreasing in k. For $k < k' < \ell$,

$$\begin{aligned} U(k,\ell) &= \sum_{a=k}^{\ell} \frac{x_a - x_k}{x_\ell - x_k} \ge \sum_{a=k'}^{\ell} \frac{x_a - x_k}{x_\ell - x_k} \\ &\ge \sum_{a=k'}^{\ell} \frac{x_a - x_{k'}}{x_\ell - x_{k'}} = U(k',\ell), \end{aligned}$$

where the second inequality follows from

$$\frac{x_a - x_k}{x_\ell - x_k} - \frac{x_a - x_{k'}}{x_\ell - x_{k'}} = \frac{(x_{k'} - x_k)(x_\ell - x_a)}{(x_\ell - x_k)(x_\ell - x_{k'})} \ge 0.$$

Next, $D(\ell, j)$ is monotone increasing in j. Indeed, for $\ell < j < j'$,

$$D(\ell, j') - D(\ell, j) \ge \sum_{a=\ell}^{j} \left(\frac{x_{j'} - x_a}{x_{j'} - x_\ell} - \frac{x_j - x_a}{x_j - x_\ell} \right),$$

and each summand is nonnegative since

$$\frac{x_{j'} - x_a}{x_{j'} - x_\ell} - \frac{x_j - x_a}{x_j - x_\ell} = \frac{(x_{j'} - x_j)(x_a - x_\ell)}{(x_{j'} - x_\ell)(x_j - x_\ell)} \ge 0.$$

The additional terms in $D(\ell, j')$ with indices $a > j$ are also nonnegative. Hence $D(\ell, j') \ge D(\ell, j)$.

Because ρ_ℓ reverses the order of the columns, the function

$$\widetilde{m}(r) = D(\ell, \rho_\ell(r))$$

is monotone decreasing in r. Thus both v and $\widetilde{m}$ are monotone decreasing, and g is concave. By Lemma 1,

$$\widetilde{G}(k,r) = g\big(v(k) + \widetilde{m}(r)\big) = \lambda h\left(\frac{U(k,\ell) + D(\ell, \rho_\ell(r))}{d} \right)$$

satisfies the quadrangle inequality. Since

$$\widetilde{M}_{k,r} = A(k) + \widetilde{G}(k,r),$$

and the row terms $A(k)$ cancel from the two sides of the quadrangle inequality, $\widetilde{M}$ also satisfies the quadrangle inequality. □

C Proof of Lemma 3

We restate the lemma for convenience.

Lemma 3. *Using Hirschberg's algorithm, the space complexity of our DP is $O(d^2)$ while the time complexity remains $O(d^2 \cdot s)$.*

Proof. Consider a subproblem with n candidate indices and r quantization values to reconstruct. A forward DP layer has one entry for each ordered pair (ℓ, j) of consecutive relevant boundary indices, so it contains $O(n^2)$ entries. By Lemma 2, for every fixed ℓ the corresponding transition matrix is Monge, and SMAWK computes the minimizing predecessor for all choices of j in $O(n)$ time. Since there are $O(n)$ choices of ℓ, one whole layer is computed in $O(n^2)$ time. The recurrence for layer i depends only on layer $i-1$, so a forward run for r layers uses $O(n^2 r)$ time and only two $O(n^2)$ arrays. The backward recurrence is symmetric and has the same bounds.

At a Hirschberg step, set $t = \lfloor r/2 \rfloor$. We run the forward recurrence for $t+1$ layers and the backward recurrence for $r-t+1$ layers, retaining only the current and previous layer in each run. We then scan the final forward and backward layers and choose a pair (ℓ^*, j^*) minimizing

$$\mathrm{dp}_{\mathrm{OPT}}(t+1, \ell, j) + \mathrm{dp}^{B}_{\mathrm{OPT}}(r-t+1, \ell, j) - \mathrm{C}[\ell, j].$$

For a fixed pair (ℓ, j), the first term is the optimal cost of the left partial solution ending at ℓ with j as its right neighbor, and the second term is the optimal cost of the right partial solution starting at j with ℓ as its left neighbor. The segment cost $\mathrm{C}[\ell, j]$ is present in both partial costs, so it is subtracted once. The entropy contributions at the two crossing quantization values are also well-defined: the left pass charges the quantization value at ℓ using its left neighbor and j, while the right pass charges the quantization value at j using ℓ and its right neighbor. Therefore the displayed expression is exactly the best total cost among solutions whose crossing adjacent pair is (ℓ, j). Minimizing over all pairs recovers the crossing pair of a globally optimal solution; otherwise, concatenating the two partial solutions for a better pair would give a strictly better feasible full solution.

After (ℓ^*, j^*) is fixed, no remaining quantization value can lie strictly between ℓ^* and j^*. The reconstruction therefore decomposes into two independent recursive subproblems: the left side, which reconstructs the quantization values up to ℓ^* with j^* fixed as the outside neighbor, and the right side, which reconstructs the quantization values from j^* onward with ℓ^* fixed as the outside neighbor. The two sets of unfixed candidate indices are disjoint. Endpoint convention choices can add only $O(1)$ fixed boundary indices to a subproblem and do not affect the asymptotic bounds.

The space bound follows immediately. During any forward or backward pass, we store only a constant number of $O(n^2)$ layers. The largest subproblem has $n \le d$, so these layers occupy $O(d^2)$ space. The precomputed arrays U, D, and C also occupy $O(d^2)$ space. Once a crossing pair is found, we store only that pair and recurse into one side; when that recursive call returns, the same work arrays are reused for the other side. The recursion stack and the output quantization values use at most $O(s) \le O(d)$ additional space, which is dominated by $O(d^2)$. Thus the peak space is $O(d^2)$.

It remains to bound the running time. For a subproblem with parameters (n, r), the two half-runs and the final scan cost

$$O(n^2(t+1)) + O(n^2(r-t+1)) + O(n^2) = O(n^2 r).$$

At recursion depth q, each active subproblem has at most $\lceil s/2^q \rceil + O(1)$ quantization values remaining. Moreover, the unfixed candidate sets of the active subproblems are disjoint subsets of the original d indices, so

$$\sum_u n_u^2 \le \left(\sum_u n_u \right)^2 \le d^2,$$

where u ranges over the subproblems at that depth and n_u is the number of entries in the subproblem. Hence the total work at depth q is at most

$$O\Big(\Big(\frac{s}{2^q}+1\Big)\,d^2\Big).$$

Summing over all $O(\log s)$ recursion depths gives

$$\sum_{q\geq 0} O\Big(\Big(\frac{s}{2^q}+1\Big)\,d^2\Big) = O(d^2 s) + O(d^2 \log s) = O(d^2 s).$$

Therefore, Hirschberg reduces the memory to $O(d^2)$ while preserving the $O(d^2 \cdot s)$ time complexity. □

D Generalizing to Arbitrary P

We now remove the simplifying assumption that $P = X$. Let $P = \langle p_1, \ldots, p_p \rangle$ be the sorted set of permissible quantization values. We assume that P contains at least one value not larger than x_1 and at least one value not smaller than x_d; otherwise, no unbiased quantizer with values in P can represent all entries of X.

For every $1 \leq a < b \leq p$, define

$$X_{a,b} = \{x \in X \mid p_a < x \leq p_b\}.$$

Thus, entries equal to the left endpoint are excluded from $U_P[a,b]$ and $D_P[a,b]$ for every interval. For the first selected quantization value, its deterministic mass is added explicitly in the initialization below. This convention is also exactly the one implemented by the prefix differences $N(p_b) - N(p_a)$, $S(p_b) - S(p_a)$, and $S_2(p_b) - S_2(p_a)$. The MSE contribution is unaffected because entries equal to a quantization value have zero variance. We define

$$C_P[a,b] = \sum_{x \in X_{a,b}} (p_b - x)(x - p_a),$$

and similarly

$$U_P[a,b] = \sum_{x \in X_{a,b}} \frac{x - p_a}{p_b - p_a}, \qquad D_P[a,b] = \sum_{x \in X_{a,b}} \frac{p_b - x}{p_b - p_a}.$$

We further write $U_P[n,n] = D_P[n,n]$ for all n. Thus, if p_k, p_ℓ, p_j are three consecutive quantization values, the expected frequency of p_ℓ is

$$U_P[k,\ell] + D_P[\ell,j].$$

Since P may contain values outside the range of X, the first and last quantization values are no longer fixed to x_1 and x_d. Let $L = \{a \in [p] \mid p_a \leq x_1\}$ be the feasible choices for the first quantization value, and let $R = \{b \in [p] \mid p_b \geq x_d\}$ be the feasible choices for the last one. For $a \in L$ and $a < j$, we initialize

$$dp^P_{\mathrm{OPT}}(2,a,j) = C_P[a,j] + \lambda \cdot h\left(\frac{n_a + D_P[a,j]}{d}\right),$$

where $n_a = |\{x \in X \mid x = p_a\}|$. This state corresponds to using p_a and p_j as the two current boundary values, while charging the entropy contribution of the left boundary p_a and leaving the contribution of p_j to be charged later.

For $i \geq 3$, the recurrence is identical to the one above after replacing input indices by permissible-value indices:

$$dp^P_{\mathrm{OPT}}(i,\ell,j) = C_P[\ell,j] + \min_{k<\ell}\Big\{ dp^P_{\mathrm{OPT}}(i-1,k,\ell) + \lambda \cdot h\Big(\frac{U_P[k,\ell] + D_P[\ell,j]}{d}\Big)\Big\}.$$

Finally, the optimal value using at most s quantization values is

$$\min_{2\le i\le s}\ \min_{\substack{\ell<j\\ j\in R}}\left\{dp^{P}_{\mathrm{OPT}}(i,\ell,j)+\lambda\cdot h\!\left(\frac{U_P[\ell,j]}{d}\right)\right\}.$$

If exactly s values are required, the outer minimum over i is replaced by $i=s$.

The quantities C_P, U_P, D_P can be computed efficiently using prefix sums over the sorted input. Let $N(t)$, $S(t)$, and $S_2(t)$ denote the number, sum, and sum of squares of entries of X that are at most t. Then for every $a<b$, letting

$$N_{a,b}=N(p_b)-N(p_a),\qquad S_{a,b}=S(p_b)-S(p_a),$$
$$S^{(2)}_{a,b}=S_2(p_b)-S_2(p_a),$$

we have

$$U_P[a,b]=\frac{S_{a,b}-p_aN_{a,b}}{p_b-p_a},\qquad D_P[a,b]=\frac{p_bN_{a,b}-S_{a,b}}{p_b-p_a},$$

and

$$C_P[a,b]=(p_a+p_b)S_{a,b}-S^{(2)}_{a,b}-p_ap_bN_{a,b}.$$

After a linear scan over X and P to build these prefix quantities, all pairwise costs are obtained in $O(p^2+d)$ time and $O(p^2)$ space.

The SMAWK acceleration from Section 4.1 continues to apply. The proof of Lemma 2 only uses the ordering of the candidate values and the monotonicity of the corresponding round-up and round-down sums; these arguments are unchanged when the candidates are $p_1,\dots,p_p$ rather than $x_1,\dots,x_d$. Likewise, the Hirschberg space reduction from Section 4.2 applies verbatim after replacing d DP indices by p permissible-value indices. Therefore, the general-P optimal algorithm runs in $O(p^2\cdot s+d)$ time and uses $O(p^2+d)$ space. When $P=X$ and the boundary values are fixed, this reduces to the formulation of Lemma 3.

E Details for Optimality Through Two-Way Time-Sharing

E.1 From Lagrange Multipliers to Entropy Constraints

The Lagrangian formulation need not find the optimal single quantizer for a given entropy constraint. For example, Let d be divisible by 2, and consider

$$X=(\underbrace{0,\dots,0}_{d/2\text{ times}},\underbrace{2,\dots,2}_{d/2\text{ times}}),$$
$$P=(0,1,2,3,4),\qquad s=2.$$

The feasible quantizers are:

Quantizer Q	Output distribution	$H(\widehat{X})$	$vNMSE$
$A=\{0,4\}$	$(3/4,\ 1/4)$	0.811	1
$B=\{0,3\}$	$(2/3,\ 1/3)$	0.918	$1/2$
$C=\{0,2\}$	$(1/2,\ 1/2)$	1.000	0

If $b=0.95$, then B is the hard-constrained optimum. However, B beats C only if

$$\lambda\ge\frac{1/2}{1-0.918296}>6,$$

while B beats A only if

$$\lambda \leq \frac{1/2}{0.918296 - 0.811278} < 5.$$

Thus, no value of λ produces B as the Lagrangian optimum.

The standard approach is to *time-share* solutions that are optimal for some value of λ. Here, the quantizer can pick a random subset of $d \cdot \alpha$ entries, where $\alpha = \frac{0.95-0.811}{1-0.811} \approx 0.74$, encode them with C, and encode the rest with A.[4] Since the entropy and vNMSE are linear in the time-sharing weight, the result has entropy $H(\widehat{X}) = 0.811 \cdot \alpha + 1 \cdot (1 - \alpha) = 0.95$ and vNMSE $0 \cdot \alpha + 1 \cdot (1 - \alpha) \approx 0.26 < 1/2$, improving over quantizer B.

E.2 Quantizing to Non-Adjacent Values in Q

In the main development, each entry $x \in X$ is stochastically quantized between $a_x = \max\{q \in Q \mid q \leq x\}$ and $b_x = \min\{q \in Q \mid q \geq x\}$. This intuitive restriction is not guaranteed to give the optimal single quantizer. Let d be divisible by 3 and consider

$$X = \big(\underbrace{0, \ldots, 0}_{d/3 \text{ times}}, \underbrace{1, \ldots, 1}_{d/3 \text{ times}}, \underbrace{2, \ldots, 2}_{d/3 \text{ times}}\big),$$

$$P = (0, 1, 2), \qquad s = 3, \qquad b = \frac{3}{2}.$$

Under the adjacent-rounding restriction, any feasible quantizer must contain 0 and 2. Hence, the only relevant choices are

$$Q = \{0, 2\} \qquad \text{and} \qquad Q = \{0, 1, 2\}.$$

The quantizer $Q = \{0, 1, 2\}$ maps all entries deterministically and yields output frequencies

$$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right),$$

so

$$H(\widehat{X}) = \log_2 3 > \frac{3}{2}.$$

Therefore, the entropy constraint forces the adjacent-rounding solution to use

$$Q = \{0, 2\}.$$

This gives output frequencies $(1/2, 1/2)$, so its entropy is $1 \leq b$. Its MSE is $d/3$. Since

$$\|X\|^2 = \frac{d}{3} \cdot 1^2 + \frac{d}{3} \cdot 2^2 = \frac{5d}{3},$$

its vNMSE is

$$\frac{d/3}{5d/3} = \frac{1}{5}.$$

If we instead allow $Q = \{0, 1, 2\}$ and permit different entries with the same value to be quantized differently, we can map half of the entries equal to 1 deterministically to 1, and stochastically quantize the

[4] By using shared randomness to pick the subset, the indices need not be communicated and the dequantizer can independently generate them.

other half between the non-adjacent values $\{0, 2\}$, with probability $1/2$ on each endpoint. This remains unbiased entrywise. The expected output frequencies are

$$\left(\frac{5}{12}, \frac{1}{6}, \frac{5}{12}\right),$$

and therefore

$$H(\widehat{X}) = -2 \cdot \frac{5}{12} \log_2 \frac{5}{12} - \frac{1}{6} \log_2 \frac{1}{6} \approx 1.48336 < \frac{3}{2}.$$

The MSE is $d/6$, and hence the vNMSE is

$$\frac{d/6}{5d/3} = \frac{1}{10}.$$

Thus, allowing non-adjacent quantization and allowing two identical entries to be treated differently can strictly improve the vNMSE while satisfying the same entropy budget when a single quantizer must be used.

E.3 Proof of Proposition 5

Under two-way time-sharing, allowing non-adjacent unbiased rounding does not improve the entropy–vNMSE frontier, even when identical input entries may be treated differently. We prove the stronger statement that every two-way time-share using arbitrary unbiased rounding is dominated by a two-way time-share using ordinary adjacent stochastic quantization.

Let $X = (x_1, \ldots, x_d) \in \mathbb{R}^d$, and assume $\|X\| > 0$. For a finite quantizer $Q \subset \mathbb{R}$, an arbitrary unbiased rounding rule is specified by probabilities

$$K_i(q) = \Pr[\widehat{x}_i = q], \qquad i \in [d],\ q \in Q,$$

satisfying

$$K_i(q) \geq 0, \qquad \sum_{q \in Q} K_i(q) = 1, \qquad \sum_{q \in Q} q K_i(q) = x_i.$$

The rule may depend on the coordinate i, so equal input values need not be treated identically. Define its expected output-frequency vector by

$$\pi_K(q) = \frac{1}{d} \sum_{i=1}^{d} K_i(q), \qquad q \in Q.$$

Thus, $\pi_K(q)$ is the expected fraction of coordinates emitted as q, and $\sum_q \pi_K(q) = 1$. Its entropy and vNMSE are

$$R(K) = H(\pi_K) = -\sum_{q \in Q} \pi_K(q) \log_2 \pi_K(q)$$

and

$$D(K) = \frac{1}{\|X\|^2} \sum_{i=1}^{d} \sum_{q \in Q} K_i(q)(q - x_i)^2,$$

where $0 \log_2 0$ is defined as 0.

A two-way time-share consists of rules K_1, K_2 for all d coordinates and a weight $\alpha \in [0, 1]$. The shared selector is independent of the data values; equivalently, one may time-share by partitioning X and using separate quantizers for each part. The quantizer used for a given entry need not be encoded, and the

two parts may be entropy-coded separately. This is the convexified time-sharing convention used in the preceding discussion. Its rate and distortion are therefore

$$R_{\mathrm{ts}} = \alpha R(K_1) + (1-\alpha)R(K_2),$$
$$D_{\mathrm{ts}} = \alpha D(K_1) + (1-\alpha)D(K_2).$$

Ordinary adjacent stochastic quantization with a sorted set $S = \{s_1 < \cdots < s_m\}$ maps each x_i to the two consecutive values of S that encompass it, with the unique probabilities that make the output mean equal to x_i.

Proposition 5. *Let $Q_1, Q_2 \subseteq P$ satisfy $|Q_1|, |Q_2| \le s$, and let K_1, K_2 be arbitrary unbiased rounding rules into Q_1, Q_2. For every $\alpha \in [0,1]$, there exist two sets $S_1, S_2 \subseteq P$, each of size at most s, and $\beta \in [0,1]$, such that ordinary adjacent stochastic quantization with S_1, S_2 satisfies*

$$(1)\ \beta R(S_1) + (1-\beta)R(S_2) \le \alpha R(K_1) + (1-\alpha)R(K_2),$$
$$(2)\ \beta D(S_1) + (1-\beta)D(S_2) \le \alpha D(K_1) + (1-\alpha)D(K_2).$$

Here, $R(S)$ and $D(S)$ denote the entropy and vNMSE of ordinary adjacent stochastic quantization with S.

The proof uses three elementary facts.

Lemma 6 (The expected output frequencies determine the vNMSE)**.** *For every unbiased rule K into Q,*

$$D(K) = \frac{d\sum_{q\in Q}\pi_K(q)q^2 - \|X\|^2}{\|X\|^2}.$$

Consequently, for fixed X and fixed output values, two unbiased rules with the same expected output-frequency vector have exactly the same vNMSE.

Proof. For each coordinate i, unbiasedness gives

$$\begin{aligned}\mathbb{E}[(\widehat{x}_i - x_i)^2] &= \sum_{q\in Q} K_i(q)(q-x_i)^2 \\ &= \sum_{q\in Q} K_i(q)q^2 - 2x_i\sum_{q\in Q} qK_i(q) + x_i^2\sum_{q\in Q} K_i(q) \\ &= \sum_{q\in Q} K_i(q)q^2 - x_i^2.\end{aligned}$$

Summing over i, exchanging the two finite sums, and dividing by $\|X\|^2$ yields

$$\begin{aligned}D(K) &= \frac{\sum_{q\in Q} q^2\sum_{i=1}^d K_i(q) - \sum_{i=1}^d x_i^2}{\|X\|^2} \\ &= \frac{d\sum_{q\in Q}\pi_K(q)q^2 - \|X\|^2}{\|X\|^2}.\end{aligned}$$

The right-hand side depends on K only through π_K. □

Lemma 7 (Every unbiased frequency vector is a mixture of adjacent-SQ frequency vectors)**.** *Fix a feasible quantizer $Q = \{q_1 < \cdots < q_m\}$, meaning $q_1 \le \min_i x_i$ and $q_m \ge \max_i x_i$. Let $\mathcal{F}_Q$ be the set of expected output-frequency vectors produced by arbitrary unbiased rules into Q. Let $\mathcal{A}_Q$ be the set of expected output-frequency vectors produced by ordinary adjacent stochastic quantization using any feasible subset $S \subseteq Q$. Then*

$$\mathcal{F}_Q = \mathrm{conv}(\mathcal{A}_Q).$$

Proof. If $m = 1$, feasibility implies that $x_i = q_1$ for every i, and both $\mathcal{F}_Q$ and $\mathcal{A}_Q$ contain the same single frequency vector. Assume henceforth that $m \geq 2$.

Every adjacent-SQ rule is an unbiased rule into Q, after assigning zero probability to values in $Q \setminus S$. Hence, $\mathrm{conv}(\mathcal{A}_Q) \subseteq \mathcal{F}_Q$, because $\mathcal{F}_Q$ is convex.

For the reverse inclusion, fix arbitrary real numbers $c(q)$ for $q \in Q$. We first minimize the linear quantity

$$\langle c, \pi_K \rangle = \sum_{q \in Q} c(q)\pi_K(q) = \frac{1}{d} \sum_{i=1}^{d} \sum_{q \in Q} c(q) K_i(q)$$

over all unbiased rules K. The constraints on different coordinates are independent, so this minimization separates over i.

Let g be the lower convex envelope of the finitely many points

$$(q_1, c(q_1)), \ldots, (q_m, c(q_m)).$$

Equivalently, g is the largest convex function on $[q_1, q_m]$ satisfying $g(q) \leq c(q)$ for every $q \in Q$. For any distribution p on Q with mean x, Jensen's inequality gives

$$g(x) = g\left(\sum_{q \in Q} p(q) q \right) \leq \sum_{q \in Q} p(q) g(q) \leq \sum_{q \in Q} p(q) c(q).$$

Thus, $g(x)$ is a lower bound on the cost of every unbiased distribution with mean x.

Let $S = \{v_1 < \cdots < v_k\} \subseteq Q$ be the q-coordinates of the vertices on the lower convex envelope. The endpoints q_1, q_m belong to S, so S is feasible. On every interval $[v_j, v_{j+1}]$, the function g is the line segment joining $(v_j, c(v_j))$ and $(v_{j+1}, c(v_{j+1}))$. Therefore, if $v_j \leq x \leq v_{j+1}$, the distribution

$$\begin{aligned} \Pr[Y = v_j] &= \frac{v_{j+1} - x}{v_{j+1} - v_j}, \\ \Pr[Y = v_{j+1}] &= \frac{x - v_j}{v_{j+1} - v_j}. \end{aligned}$$

has mean x and expected cost exactly $g(x)$. This is precisely ordinary adjacent stochastic quantization with the set S. Applying it to every coordinate attains the minimum of $\langle c, \pi_K \rangle$. Hence, for every c,

$$\min_{\pi \in \mathcal{F}_Q} \langle c, \pi \rangle = \min_{a \in \mathcal{A}_Q} \langle c, a \rangle. \tag{7}$$

It remains to show that (7) implies $\mathcal{F}_Q \subseteq \mathrm{conv}(\mathcal{A}_Q)$. Suppose instead that some $\pi \in \mathcal{F}_Q$ lies outside the compact convex set $C = \mathrm{conv}(\mathcal{A}_Q)$. Choose $y \in C$ minimizing $\|\pi - y\|$ and set $c = y - \pi$. For every $a \in C$, convexity of C implies that $y + t(a - y) \in C$ for $t \in [0, 1]$. Since $t = 0$ minimizes

$$\|\pi - (y + t(a - y))\|^2,$$

the right derivative at 0 is nonnegative, which gives

$$(y - \pi) \cdot (a - y) \geq 0.$$

Thus, $c \cdot a \geq c \cdot y$ for every $a \in C$, while

$$c \cdot \pi = c \cdot y - \|c\|^2 < c \cdot y.$$

Therefore,

$$\min_{f \in \mathcal{F}_Q} c \cdot f \le c \cdot \pi < \min_{a \in C} c \cdot a$$
$$= \min_{a \in \mathcal{A}_Q} c \cdot a,$$

contradicting (7). Hence, $\mathcal{F}_Q = \mathrm{conv}(\mathcal{A}_Q)$. □

Lemma 8 (A rate-constrained mixture needs at most two quantizers)**.** *Let (R_j, D_j), $j = 1, \dots, N$, be finitely many rate–vNMSE pairs. If some mixture of these pairs has rate at most b, then the minimum vNMSE among all mixtures with rate at most b is attained by a mixture supported on at most two pairs.*

Proof. Consider the finite-dimensional linear program

$$\min_{w_1, \dots, w_N} \sum_{j=1}^{N} w_j D_j$$

subject to

$$\sum_{j=1}^{N} w_j = 1, \qquad \sum_{j=1}^{N} w_j R_j \le b, \qquad w_j \ge 0. \tag{8}$$

Its feasible set is compact, so an optimum exists. An optimum may be chosen to be an extreme point of the feasible set. Indeed, if an optimal point is a nontrivial convex combination of two feasible points, linearity implies that both points are also optimal; repeating within the resulting lower-dimensional face reaches an extreme optimal point.

Let w be an extreme feasible point, and suppose that it has at least three positive coordinates. The vectors $(1, R_j) \in \mathbb{R}^2$ corresponding to those positive coordinates are linearly dependent. Hence, there is a nonzero vector z, supported only on those coordinates, such that

$$\sum_j z_j = 0, \qquad \sum_j z_j R_j = 0.$$

For sufficiently small $\varepsilon > 0$, both $w + \varepsilon z$ and $w - \varepsilon z$ are nonnegative. They preserve both sums in (8), so they are distinct feasible points, and

$$w = \frac{1}{2}(w + \varepsilon z) + \frac{1}{2}(w - \varepsilon z).$$

This contradicts extremality. Therefore, an extreme feasible point, and in particular an extreme optimum, has at most two positive coordinates. □

Proof of Proposition 5. The existence of an unbiased rule K_t into Q_t implies that every x_i lies in the convex hull of Q_t; hence Q_t is feasible. Apply Lemma 7 separately to K_1 and K_2. For each $t \in \{1, 2\}$, there are feasible subsets $S_{t,r} \subseteq Q_t$ and weights $\theta_{t,r} \ge 0$ with $\sum_r \theta_{t,r} = 1$ such that

$$\pi_{K_t} = \sum_r \theta_{t,r} \pi_{S_{t,r}}. \tag{9}$$

By Lemma 6 and the linearity of its right-hand side in π,

$$D(K_t) = \sum_r \theta_{t,r} D(S_{t,r}). \tag{10}$$

Entropy is concave. To see this directly, the function $h(u) = -u \log_2 u$, with $h(0) = 0$, satisfies $h''(u) = -1/(u \ln 2) < 0$ for $u > 0$. Therefore, Equation (9) gives

$$R(K_t) = H(\pi_{K_t}) \geq \sum_r \theta_{t,r} H(\pi_{S_{t,r}}) = \sum_r \theta_{t,r} R(S_{t,r}). \tag{11}$$

Combining the decompositions for the two original arms with weights

$$w_{1,r} = \alpha\theta_{1,r}, \qquad w_{2,r} = (1-\alpha)\theta_{2,r}$$

produces a finite mixture of adjacent-SQ quantizers whose MSE equals the original two-way vNMSE by Equation (10), and whose rate is no larger than the original two-way rate by Equation (11). Every quantizer in this finite collection is a subset of Q_1 or Q_2, so it has at most s values.

This finite mixture is only an intermediate existence certificate. Apply Lemma 8 to its finitely many adjacent-SQ rate–vNMSE pairs, using the original rate R_{ts} as the budget. The resulting optimal mixture uses at most two adjacent-SQ quantizers, has rate at most R_{ts}, and has vNMSE no larger than the intermediate mixture, hence no larger than D_{ts}. This proves the proposition. □

Consequently, there is no instance in this model where a two-way time-share using non-adjacent unbiased rounding has strictly smaller vNMSE than every adjacent-only solution that may time-share between at most two quantizers.

For the example above, adjacent time-sharing between $\{0, 1, 2\}$ and $\{0, 2\}$ is already better than the proposed non-adjacent rule. Put weight

$$\alpha = \frac{3/2 - 1}{\log_2 3 - 1} \approx 0.855$$

on $\{0, 1, 2\}$ and the remaining weight on $\{0, 2\}$. The average entropy is exactly $3/2$, while the vNMSE is

$$(1-\alpha)\frac{1}{5} \approx 0.029 < \frac{1}{10}.$$

Thus, the existing example does not separate non-adjacent rounding from adjacent rounding once two-way time-sharing is admitted.

F Evaluation Details

F.1 Protocol and Rate Accounting

All experiments use BF16 inputs and report the scale-free vNMSE defined in Section 2. To compare methods at a common storage budget, we include both the ideal entropy-coded payload and the per-level metadata:

$$R_{\mathrm{tot}}(Q, X) = H(\widehat{X}) + \frac{(16+16)|Q|}{d} = H(\widehat{X}) + \frac{32|Q|}{d} \quad \text{bits/value.}$$

For each active quantization value, the first 16 bits store its BF16 reconstruction value and the second 16 bits model its entropy descriptor, such as a quantized frequency/CDF or code-length entry. The ordering of Q supplies the symbol-to-value association, so we do not charge for a separate permutation. This remains an ideal rate: $H(\widehat{X})$ is the ideal payload rate, and we exclude global framing and finite-stream redundancy. At a target rate, we average repeated rates arithmetically and vNMSEs geometrically, interpolate vNMSE in log space between adjacent measured points, and never extrapolate. Every reported method–tensor curve brackets all six target rates.

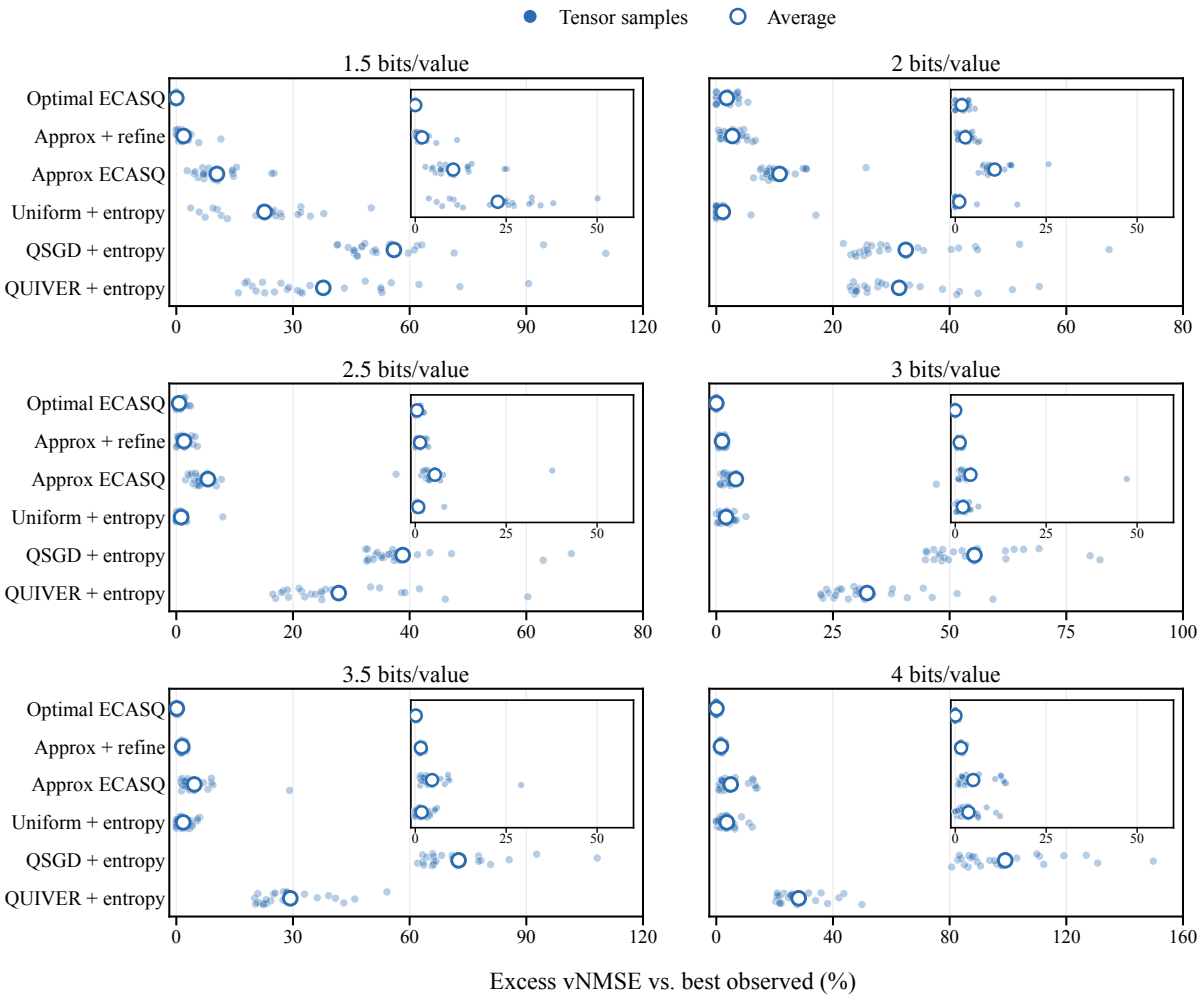


Figure 5: Matched-rate vNMSE over 25 BF16 real-model weight tensors. Subfigures show 1.5, 2, 2.5, 3, 3.5, and 4 bits/value in reading order. Filled circles are individual tensors and the larger white-centered circle is their arithmetic mean. Every full subfigure uses a linear horizontal axis sized to its observed range; its inset magnifies Optimal ECASQ, Approx ECASQ with refinement, Approx ECASQ, and uniform quantization with entropy coding on a common linear scale within this figure. Values are relative to the best observed method for the same tensor and total rate, and no method is extrapolated.

We compare Optimal ECASQ, Approx ECASQ, Approx ECASQ followed by five refinement rounds, QUIVER with entropy coding, QSGD with entropy coding, and uniform stochastic quantization with entropy coding. The synthetic rate–distortion study uses vectors of length $d = 16{,}384$, five seeds, at most $s = 64$ quantization values, and Normal, Lognormal, Student-t, and sparse-outlier distributions. We sweep 25 logarithmically spaced Lagrange multipliers from 10^{-6} through 10^{2} for ECASQ. QUIVER uses requested codebook sizes from 2 through 256 in powers of two; QSGD and uniform quantization continue through 4096. We then compare at $R_{\mathrm{tot}} \in \{1.5, 2, 2.5, 3, 3.5, 4\}$ bits/value.

The real-model study contains 100 fixed samples from five model families: Qwen2.5 0.5B and 1.5B, Gemma 3 1B IT, Llama 3.2 1B, and Phi-3.5 Mini. It comprises 25 weight matrices, 45 activations, and 30 KV-cache tensors. Each sample contains $d = 16{,}384$ entries. Activations and KV caches come from the first, middle, and last decoder blocks; the weights cover embedding, attention-input, attention-output, MLP-input, and MLP-output families. Optimal ECASQ and Approx ECASQ ordinarily use $s = 64$ and

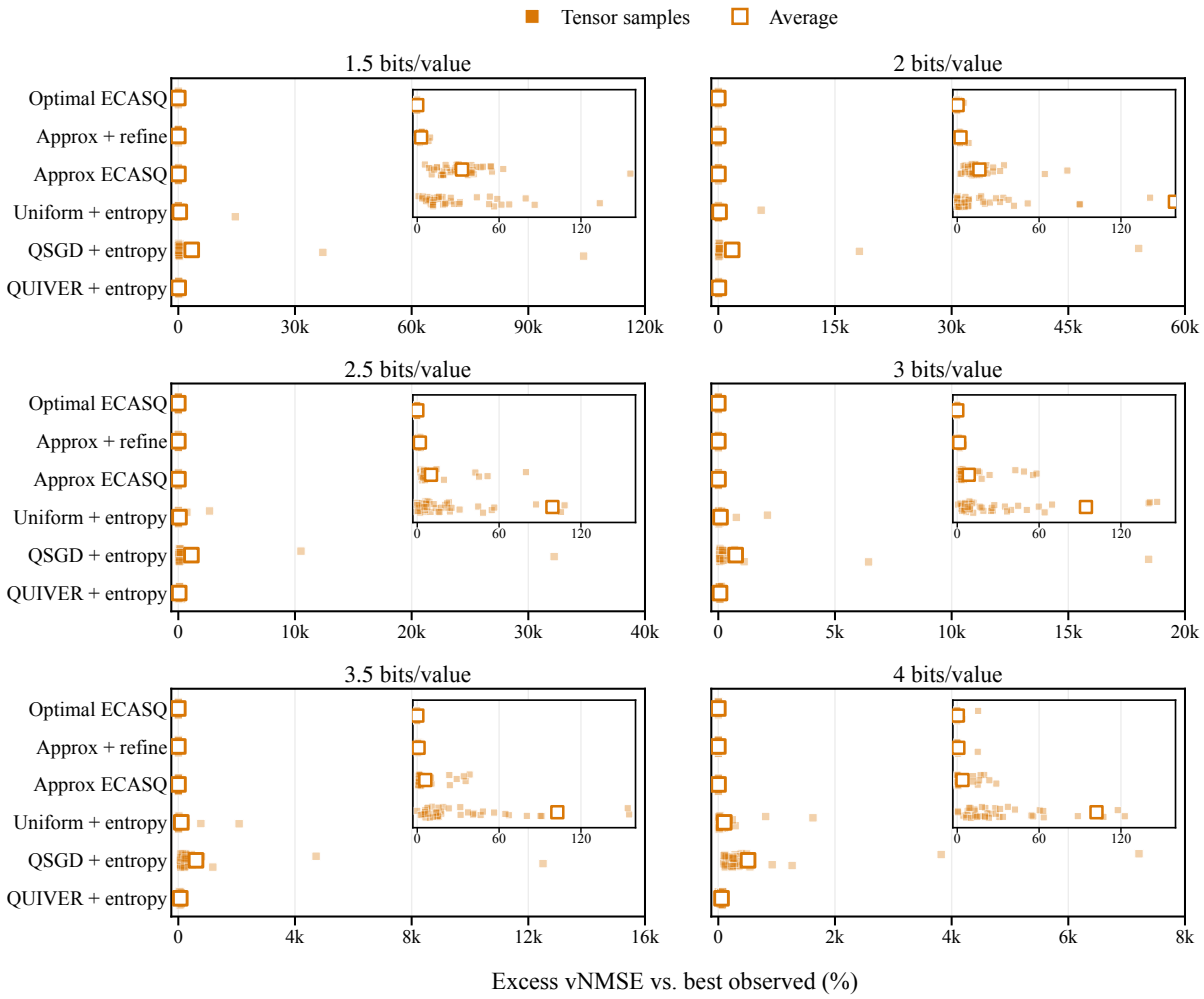


Figure 6: Matched-rate vNMSE over 45 BF16 real-model activation tensors. Subfigure order and axis conventions match Figure 5. Filled squares are individual tensors and the larger white-centered square is their arithmetic mean. Each inset uses a common linear scale within this figure and magnifies the three ECASQ variants together with uniform quantization with entropy coding.

$\lambda \in \{10^{-6}, 10^{-5}, \ldots, 10^{2}\}$; the baseline codebook-size grids match the synthetic study. To bracket every target without extrapolation, we evaluate Optimal ECASQ on all 25 weight samples, add an $s = 128$, $\lambda = 0$ endpoint for one high-rate activation curve, extend fixed-codebook baselines through 16,384 quantization values where needed, and use a 65,536-bin QUIVER approximation grid for three extreme-range activations. Each activation/KV calibration run uses four 128-token sequences. The plots use the lowest measured/interpolated vNMSE for each tensor as a neutral "best observed" reference. The deterministic real-model samples do not support seed-wise confidence intervals.

F.2 Matched-Rate vNMSE

Figures 5 to 7 give the complementary real-model study at every target rate, with all 100 tensors present for every method. Refined Approx ECASQ remains close to the best observed method. At 3 bits/value its average and maximum excess vNMSE are 1.35% and 4.8%, respectively. Unrefined Approx ECASQ averages 8.12%, lower than uniform, QUIVER, and QSGD at 44.35%, 54.9%, and 390.57%, respectively.

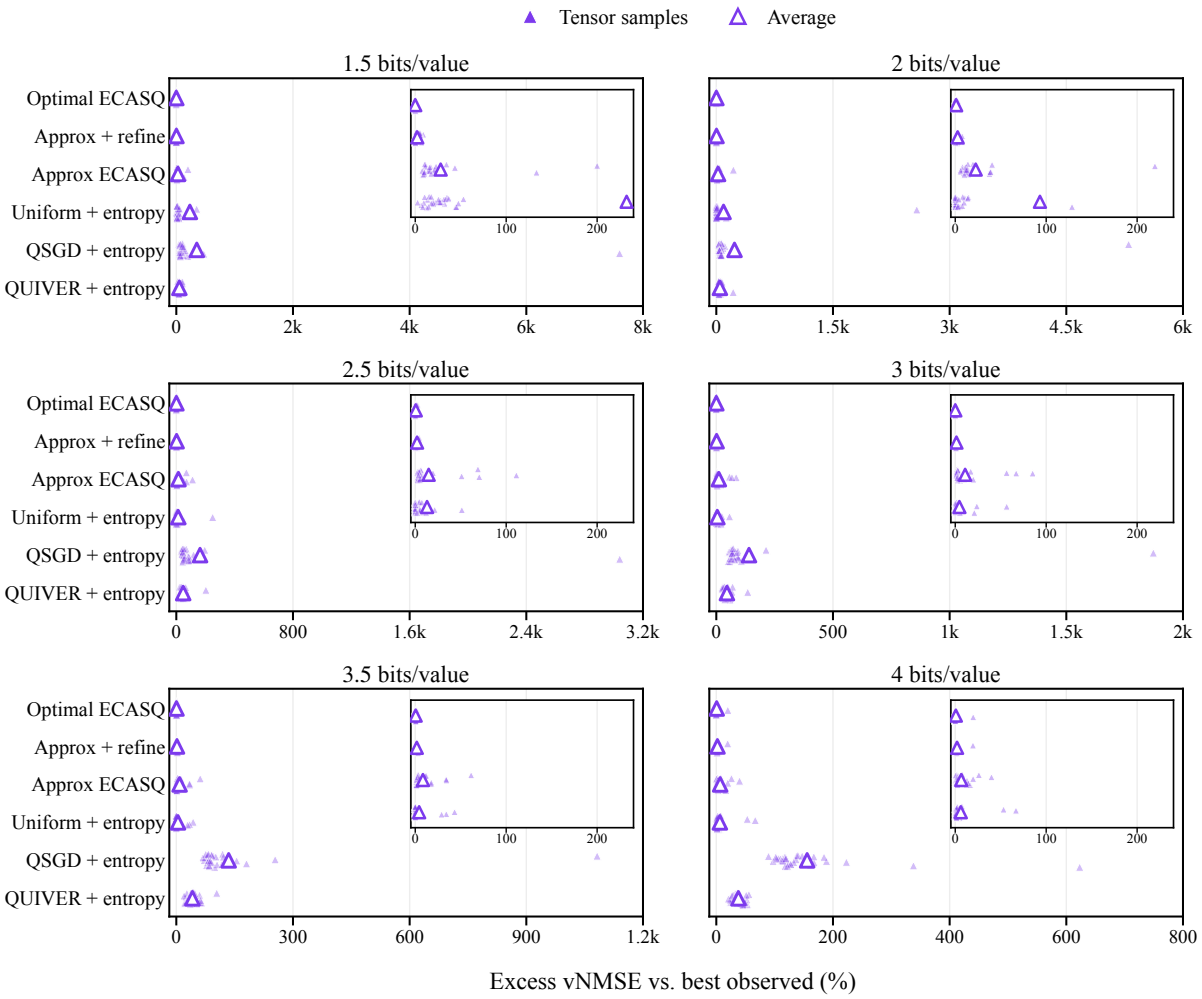


Figure 7: Matched-rate vNMSE over 30 BF16 real-model KV-cache tensors. Subfigure order and axis conventions match Figure 5. Filled triangles are individual tensors and the larger white-centered triangle is their arithmetic mean. Each inset uses a common linear scale within this figure and magnifies the three ECASQ variants together with uniform quantization with entropy coding.

F.3 Paired Uniform-Grid Control

To isolate the benefit of selecting Q from a common permissible alphabet, we first run uniform quantization on each real-model tensor. At each target rate, we retain the two measured uniform min–max grids P that bracket the target, then run all three ECASQ variants on those exact stored grids with $s_{\max} = |P|$. We convexify in $(R_{\text{tot}}, \text{vNMSE})$ space using linear-vNMSE time sharing; every component comparison therefore has the same permissible grid and maximum number of quantization values, and the rate includes the 32-bit per-level metadata overhead.

Optimal ECASQ is limited to grids with at most 256 stored quantization values. Applying this limit as a single all-rate cohort leaves 25 weight tensors, 30 activation tensors, and 29 KV-cache tensors. We omit the remaining 15 activation tensors and one KV-cache tensor from every method and target rather than report partial coverage.

Across the 18 tensor-category–rate cells in Figure 8, refined Approx ECASQ, unrefined Approx ECASQ, and uniform quantization have mean excess vNMSEs of 3.35%, 4.41%, and 12.25%, respectively. Refine-

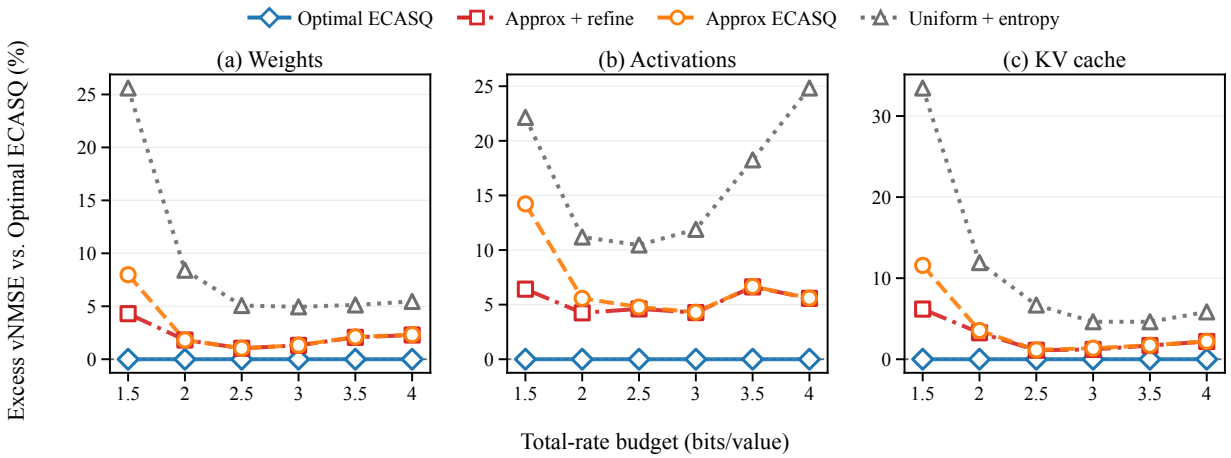


Figure 8: Matched-rate comparison on paired uniform permissible grids. The subfigures show weights, activations, and KV-cache tensors. Curves report the arithmetic mean of each tensor's excess vNMSE relative to Optimal ECASQ at the same total-rate budget. Approx ECASQ with refinement stays close to Optimal ECASQ even when all methods use the stored grids selected by the uniform baseline; lower is better.

ment is within 6.62% of Optimal ECASQ in every cell, while the uniform baseline reaches 33.46% excess.

The synthetic study provides a controlled comparison in which Optimal ECASQ is available throughout. The main-paper Figure 3 selects the fully covered 2, 3, and 4 bits/value targets. Across those 12 distribution–rate points, five refinement rounds reduce mean excess vNMSE from 4.97% to 1.37%, with a 4.06% maximum on the five-seed aggregate curves. Unrefined Approx ECASQ beats QSGD and QUIVER at all 12 points and uniform quantization at 11 of 12. Across all four distributions and six target rates, refinement reduces the mean excess from 6.06% to 1.4% and the maximum from 23.83% to 4.06% relative to Optimal ECASQ. Figure 9 shows all six matched-rate comparisons. Ten high-cardinality QUIVER runs failed; the no-extrapolation rule omits only unsupported target points rather than extending incomplete curves.

F.4 Approximation and Refinement Reality vs. Guarantee

Figure 10 isolates the entropy-surrogate effect. For an Approx ECASQ codebook, replacing the optimized surrogate $H(I, B)$ with the exact output entropy $H(\widehat{X})$ preserves vNMSE and saves exactly $H(B \mid \widehat{X})$ bits/value; subfigure (a) shows how this saving varies with the Approx surrogate budget b. All 100 interpolated distribution/seed/budget checks satisfy the empirical one-bit guarantee, and $H(B \mid \widehat{X})$ remains within the proved one-bit bound.

Figure 11 reports refinement separately. Starting from Approx ECASQ, five rounds reduce the median objective gap to 0.12% (maximum 0.58% over the 20 distribution/seed cases), with most of the improvement occurring in the first few rounds. Runtime grows only slightly from 10 to 20 requested rounds because the round count is an upper bound: refinement terminates as soon as a complete round fails to decrease the objective, so most runs have converged before reaching 20. The plotted timing summary takes the running maximum for each distribution/seed trajectory before computing the median, making the end-to-end curve monotone while suppressing small measurement fluctuations between independent runs.

F.5 Runtime and Objective Scaling

We measure the cached Approx implementation on a 12-core Apple M2 Max with 32 GB of memory, using macOS 15.7.2 (arm64), Python 3.9.6, PyTorch 2.8.0, and eight CPU threads. histogram construction and

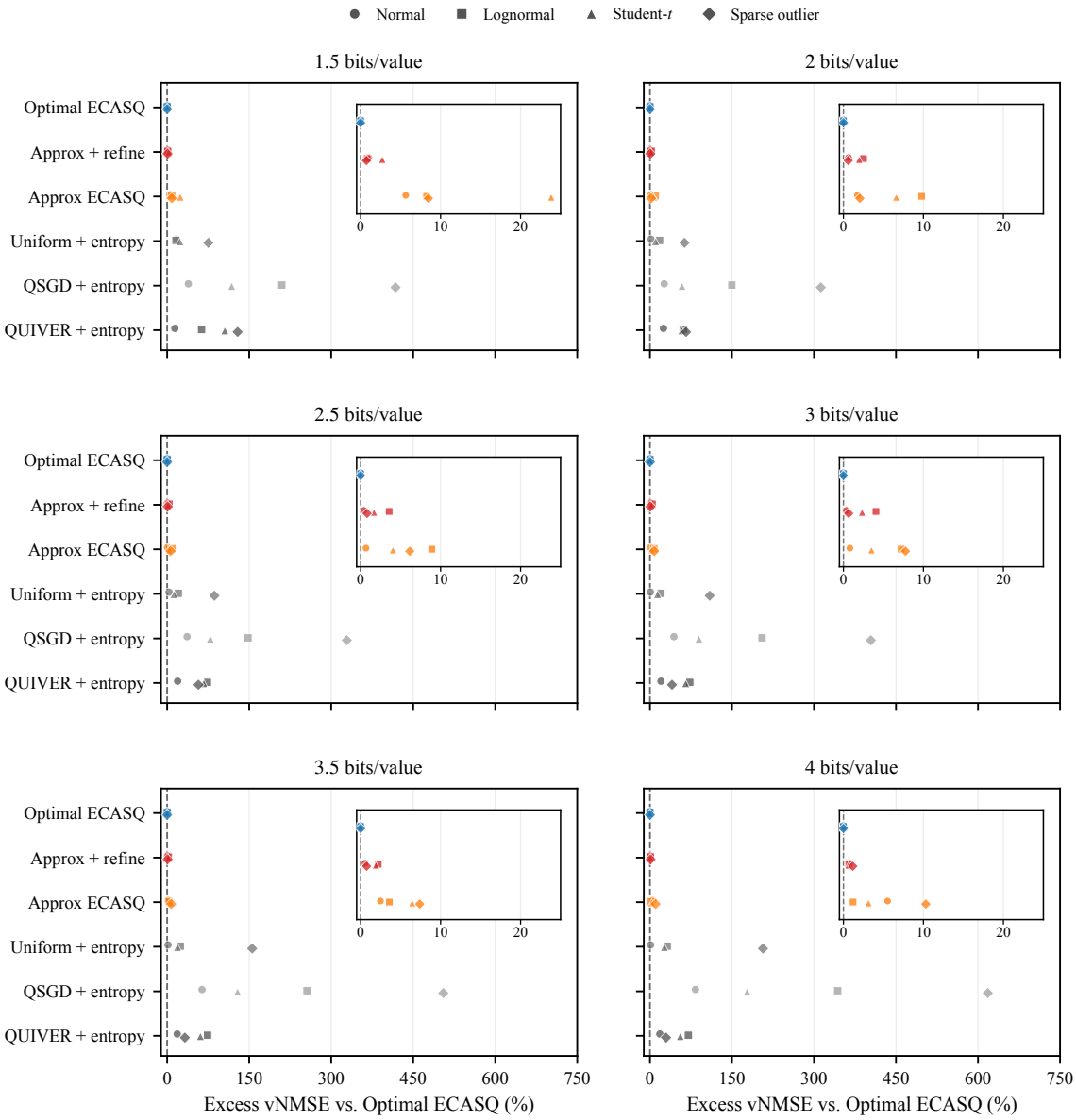
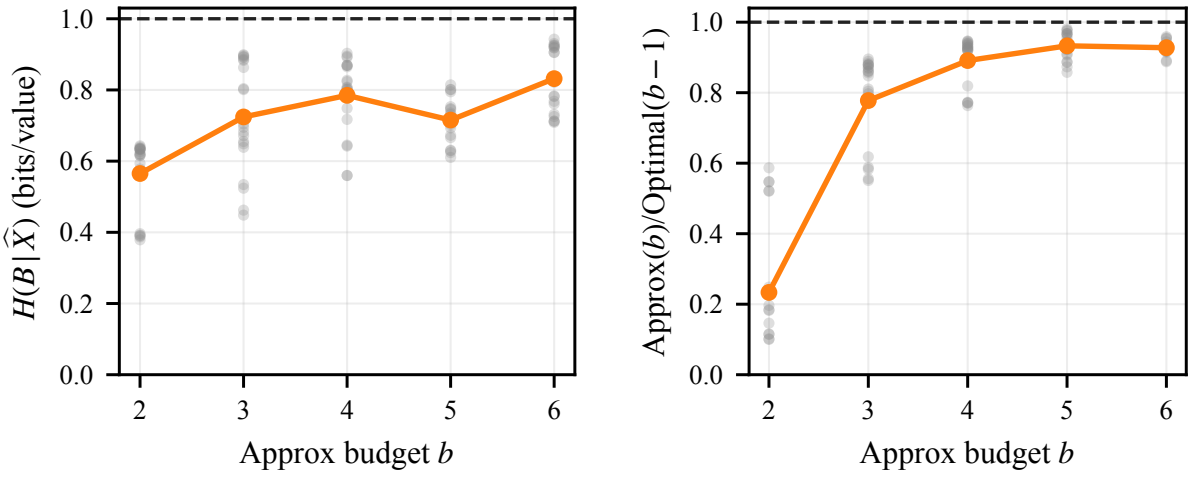


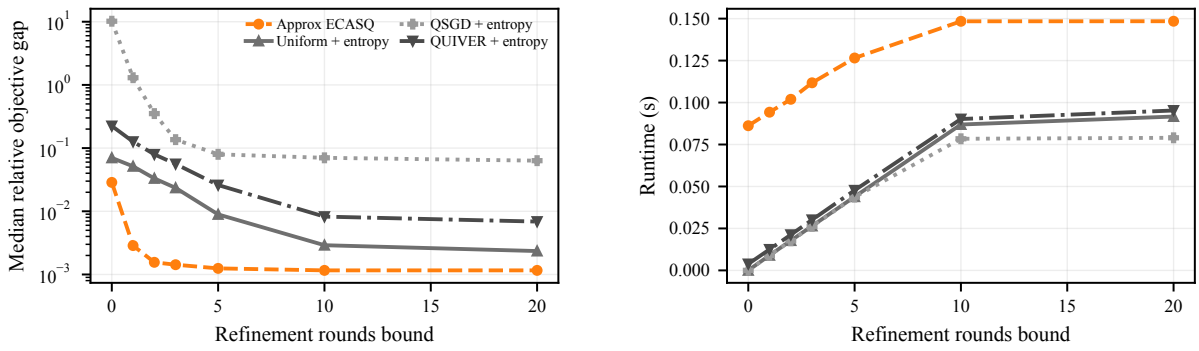
Figure 9: Synthetic matched-rate comparison at six total rates, using the same method colors and distribution markers as Figure 3. Circles, squares, triangles, and diamonds denote Normal, Lognormal, Student-t, and sparse-outlier distributions, respectively. Insets magnify the first three rows on a shared 0–25% scale. Total rate includes output entropy and the 32-bit per-level metadata. Smaller values nearer zero are better.



(a) Exact-entropy savings by Approx budget.

(b) Empirical one-bit guarantee.

Figure 10: Approximation diagnostics over the 20 distribution–seed cases at each budget. Gray points show individual cases. Orange circles show the arithmetic mean in subfigure (a) and the geometric mean in subfigure (b). The dashed horizontal line marks the proved upper bound of one. Subfigure (a) reports the bits saved by exact output-entropy accounting for the same Approx ECASQ codebook selected under surrogate budget b, at unchanged vNMSE: $H(I, B) - H(\widehat{X}) = H(B \mid \widehat{X})$. Subfigure (b) reports the vNMSE ratio between the surrogate-feasible Approx solution at budget b and Optimal ECASQ at budget $b - 1$. Both subfigures linearly interpolate the lower convex envelopes of the measured Lagrangian sweeps, corresponding to whole-vector time sharing between neighboring retained frontier points.



(a) Convergence to Optimal ECASQ.

(b) End-to-end quantizer time.

Figure 11: Refinement from four initializations. Subfigure (a) shows the median objective gap to Optimal ECASQ. Subfigure (b) shows median end-to-end runtime, including initialization and all executed refinement rounds; requested rounds are an upper bound because refinement stops after a round without objective improvement. To reflect nondecreasing work despite run-to-run timing noise, each trajectory uses its running-maximum time before aggregation.

method-specific setup, optimization, and refinement; it excludes random-vector generation, metric evaluation, and the shared final codebook cast. Each appendix point is the mean over five independently generated vectors, with a two-sided 95% Student-t confidence interval. Optimal ECASQ and the two approximate variants receive the same weighted BF16 histogram.

The main-paper runtime result in Figure 4 aggregates the four distribution-specific means; its error bars show their min–max range rather than treating the distributions as exchangeable samples. At $d = 262{,}144$ and $s = 64$, the pooled mean runtimes are 6.39 s for Optimal ECASQ, 0.26 s for Approx, and 0.33 s for refined Approx, corresponding to $24.4\times$ and $19.2\times$ speedups. Across the four distributions, the respective speedup ranges are 13.4–34.3× and 8.7–28.3×. These timing experiments hold $\lambda = 0.1$ fixed. The lower subfigures therefore report the corresponding Lagrangian objective $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$ rather than a matched-rate distortion. The fast cached implementation is distinct from the streamed, linear-space variant analyzed in the main text. Figure 12 reports the $s = 16$ codebook-size setting, Figure 13 gives the full per-distribution scaling curves for the main-paper setting, and Figure 14 reports the $s = 256$ setting.

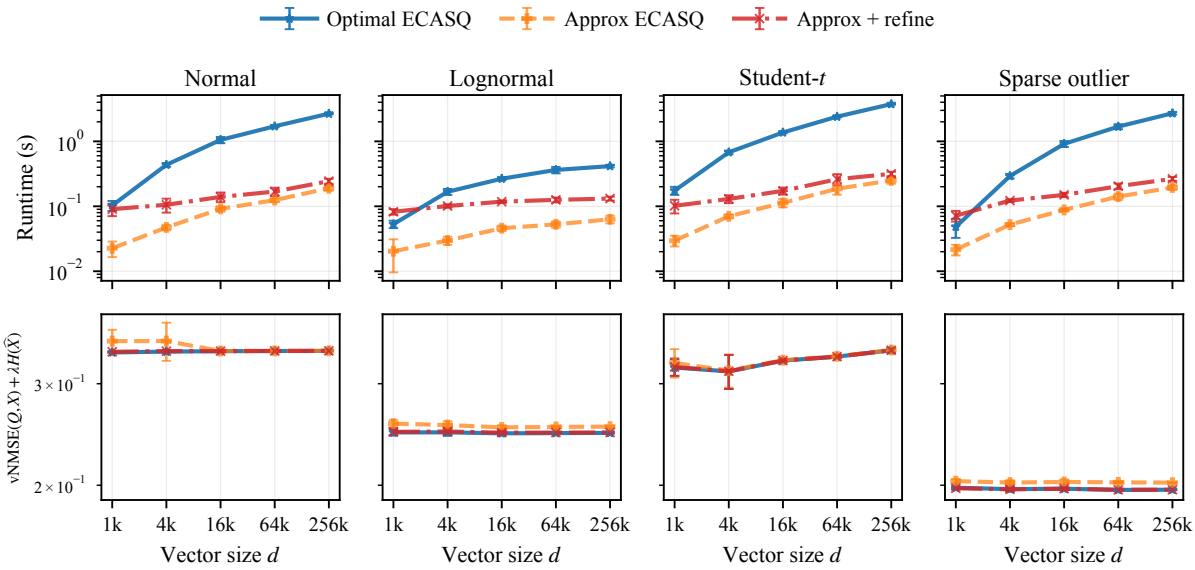


Figure 12: Runtime and objective scaling for BF16 inputs with at most $s = 16$ quantization values and fixed $\lambda = 0.1$. Points show the mean and two-sided 95% Student-t confidence interval over five seeds. The lower subfigures report $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$. All runtime subfigures share one y-axis range, and all objective subfigures share another.

F.6 Effect of the Permissible Alphabet P

The general-P algorithm in Section D permits quantization values that need not occur in the input. We therefore compare the observed support $P_X = \{x_i : i \in [d]\}$ with smaller candidate alphabets. For BF16 vectors, the alternatives are weighted empirical quantiles and BF16-rounded uniform grids between the observed minimum and maximum; both include the extrema. We request $p \in \{16, 32, 64, 128, 256\}$ candidates, although duplicate removal can make the actual value of $|P|$ smaller. We use the same four distributions and five seeds as the synthetic study, with $d = 16{,}384$, at most $s = 16$ quantization values, and fixed $\lambda = 0.1$. Every point is computed with the optimal Hirschberg grid DP. Runtime includes histogram and candidate-set construction and optimization, but excludes vector generation and metric serialization.

Uniform grids give the strongest overall tradeoff. With 256 requested candidates, their mean objective

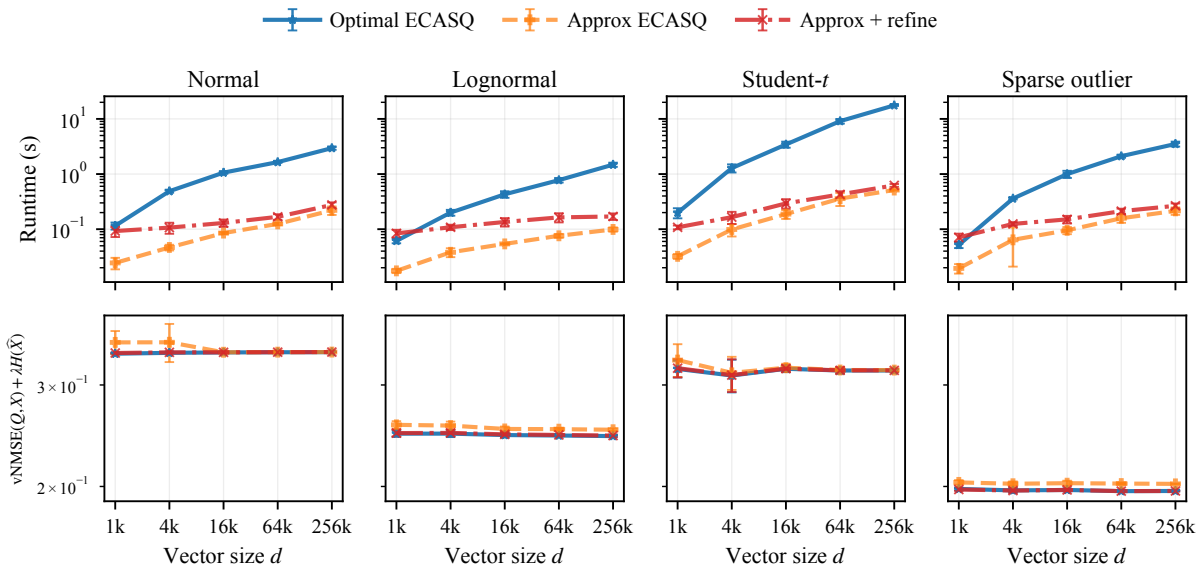


Figure 13: Runtime and objective scaling for BF16 inputs with at most $s = 64$ quantization values and fixed $\lambda = 0.1$. Points show the mean and two-sided 95% Student-t confidence interval over five seeds. The lower subfigures report $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$. All runtime subfigures share one y-axis range, and all objective subfigures share another.

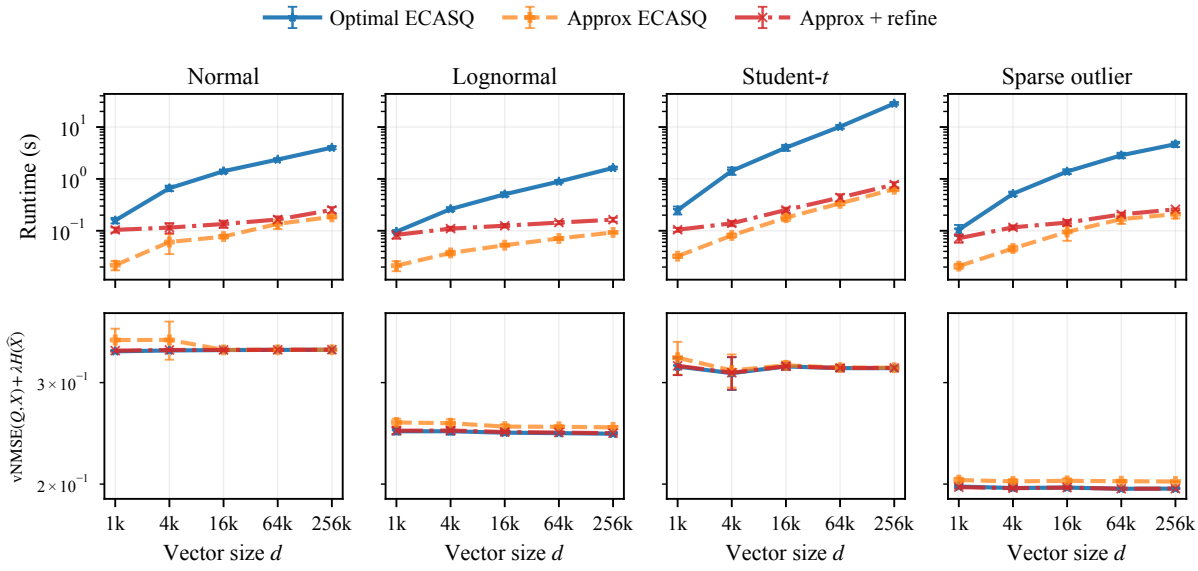


Figure 14: Runtime and objective scaling for BF16 inputs with at most $s = 256$ quantization values and fixed $\lambda = 0.1$. Points show the mean and two-sided 95% Student-t confidence interval over five seeds. The lower subfigures report $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$. All runtime subfigures share one y-axis range, and all objective subfigures share another.

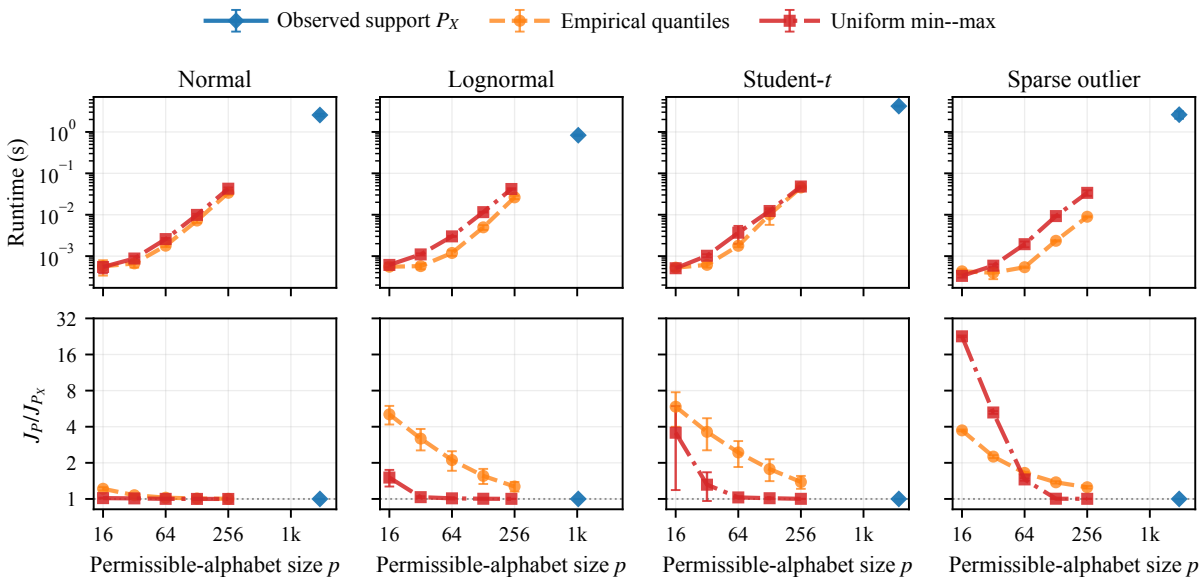


Figure 15: Runtime–quality tradeoff induced by the permissible alphabet P. Each input is a BF16 vector $X \in \mathbb{R}^d$ with $d = 16{,}384$; columns show Normal, Lognormal, Student-t, and sparse-outlier inputs. We use at most $s = 16$ quantization values, $\lambda = 0.1$, requested candidate counts $p \in \{16, 32, 64, 128, 256\}$. The upper row reports runtime, including histogram construction, candidate-set construction, and optimization, but excluding vector generation and metric serialization. The lower row reports J_P/J_{P_X}, where $J_P = \mathrm{vNMSE}(Q_P, X) + \lambda H(\widehat{X})$. Points and error bars show the mean and two-sided 95% Student-t confidence interval. The horizontal axis uses the actual candidate count after duplicate removal.

is within 0.25% of the observed-support result for every distribution while reducing runtime by 20–88×. Empirical quantiles are competitive for Normal inputs, but at 256 candidates their objective penalty is 26.9%, 38.5%, and 25.2% on Lognormal, Student-t, and sparse-outlier inputs, respectively. Equal-mass quantiles devote few candidates to tails and isolated extremes, whereas the uniform min–max grid preserves range coverage.

F.7 Entropy–vNMSE Frontiers and Time Sharing

We next measure how much two-way time sharing can improve the entropy-constrained frontier for the four synthetic distributions. Write $D(Q) = \mathrm{vNMSE}(Q, X)$ and $R(Q) = H(\widehat{X}_Q)$. For a maximum codebook size s and retained set of multipliers Λ, define the best retained feasible single quantizer and the dual lower bound

$$D_{\mathrm{single,feasible}}(b) = \min_{\lambda \in \Lambda:\, R(Q_\lambda) \le b} D(Q_\lambda),$$
$$\phi(\lambda) = \min_{Q \subseteq P_X,\, |Q| \le s} \{D(Q) + \lambda R(Q)\},$$
$$L(b) = \max_{\lambda \in \Lambda} \{\phi(\lambda) - \lambda b\}.$$

Every retained $\phi(\lambda)$ is computed by Optimal ECASQ, which globally minimizes the Lagrangian objective. For any deterministic quantizer or time-share with rate at most b, its distortion is at least $\phi(\lambda) - \lambda b$. Consequently,

$$L(b) \le D^\star_{\mathrm{TS}}(b) \le D^\star_{\mathrm{det}}(b) \le D_{\mathrm{single,feasible}}(b),$$

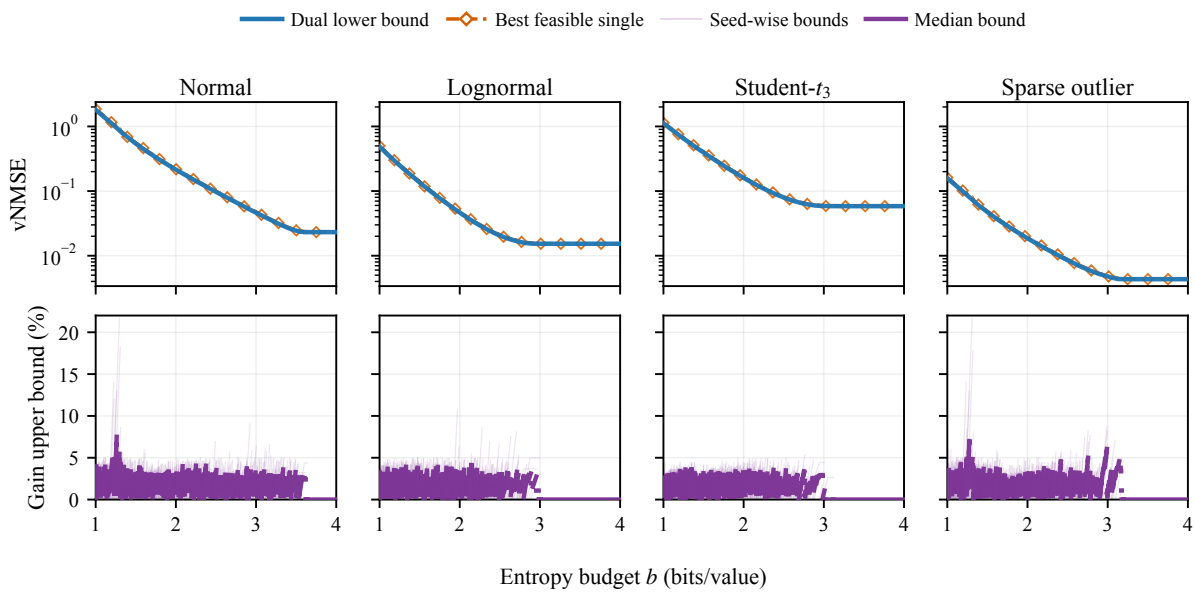


Figure 16: Entropy-constrained vNMSE and an upper bound on the benefit of two-way time sharing. Columns show BF16 Normal, Lognormal, Student-t, and sparse-outlier vectors with $d = 16{,}384$, $P = P_X$, at most $s = 16$ quantization values, and entropy budgets $b \in [1, 4]$ bits/value, sampled every 0.001 bits and augmented with retained feasible-frontier breakpoints. For each input, Optimal ECASQ solutions supported by $\mathrm{vNMSE}(Q, X) + \lambda H(\widehat{X})$ are adaptively refined from $\lambda = 0$ through an endpoint attaining at most one bit/value. Curves in the upper row are arithmetic means over the five seeds: solid lines show the dual lower bound $L(b)$, while dashed steps and open diamonds show the best retained feasible distortion $D_{\mathrm{single,feasible}}(b)$. For each seed, these quantities give the relative-gain upper bound $U(b) = 100[1 - L(b)/D_{\mathrm{single,feasible}}(b)]$ shown in the lower row. Thin lines are the five seeds and the thick line is their pointwise median. All lower subfigures here and in Figures 17 and 18 use the same 0–22% linear scale. Entropy is $H(\widehat{X})$ and excludes reconstruction-value and entropy-model metadata, global framing, and finite-stream redundancy.

where $D^{\star}_{\mathrm{TS}}$ and $D^{\star}_{\mathrm{det}}$ are the optimal time-shared and deterministic frontiers. The lower subfigures therefore report the upper bound

$$U(b) = 100 \left(1 - \frac{L(b)}{D_{\mathrm{single,feasible}}(b)} \right)$$

on the percentage improvement that time sharing can provide over the optimal deterministic single quantizer. We use output entropy here, rather than the metadata-inclusive total rate, because this is the constraint convexified by the time-sharing result in Section 7.

For numerical conservatism, before forming $L(b)$ we reduce every stored Lagrangian objective ϕ by $\max\{2{\times}10^{-7}, 2{\times}10^{-5}|\phi|\}$, the solver's objective-validation tolerance. This can only lower $L(b)$ and hence loosen the reported upper bound.

We adaptively refine the supported frontier. Starting from $\lambda = 0$ and an endpoint with entropy at most one bit/value, each adjacent discovered pair is queried at the multiplier given by its supporting-line slope. An interval is closed when this query finds no new supported point, or when the relative vNMSE change between its endpoints is at most 5%. Above the entropy of the $\lambda = 0$ endpoint, the curves are extended constantly because that endpoint already globally minimizes distortion. On the 0.001-bit budget grids, for $s = 16, 64, 256$, respectively, $U(b)$ has medians $1.16\%, 1.59\%, 1.96\%$, 95th percentiles $4.07\%, 5.03\%, 6.12\%$, and is at most

5% in 98.2%, 94.9%, 91.4% of the 60k, 100k, and 140k distribution/seed/budget checks. The maximum is 21.7% for each quantization-value budget and occurs on a narrow low-rate interval. Thus, time sharing completes the theory, while the exact Lagrangian objectives show that its possible improvement is small at most tested budgets. This upper bound relies on global optimality of the Lagrangian solves: the objective of an arbitrary feasible or approximate solution is an upper bound on $\phi(\lambda)$ and cannot be substituted into $L(b)$.

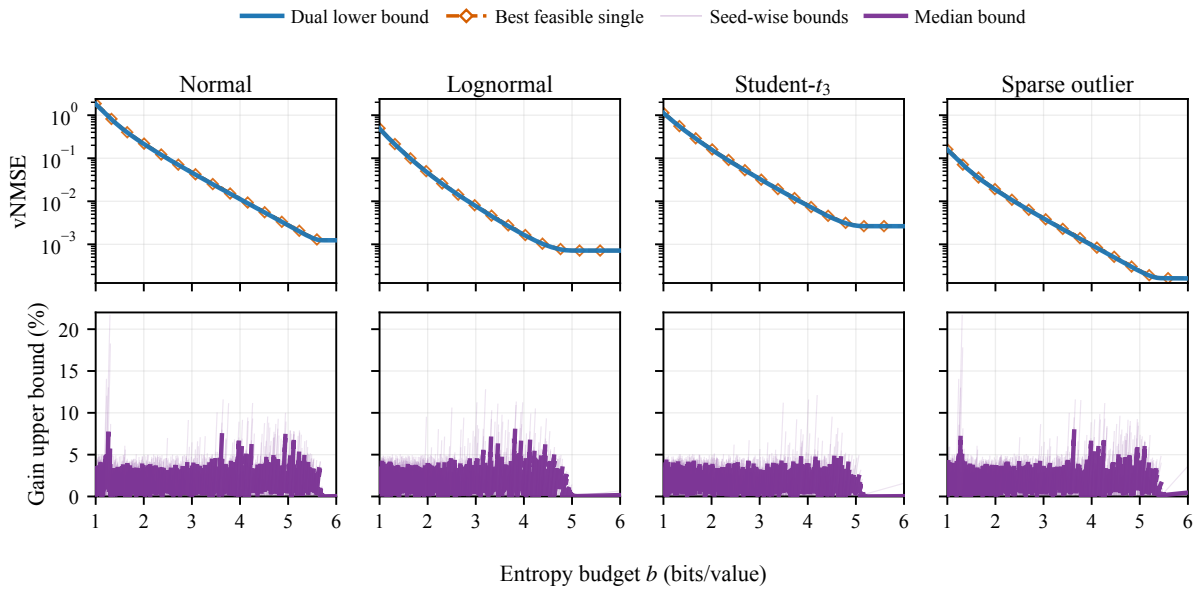


Figure 17: Entropy-constrained vNMSE and the time-sharing gain upper bound with at most $s = 64$ quantization values. The setup and curve encodings match Figure 16: columns show the four BF16 synthetic distributions with $d = 16{,}384$, $P = P_X$; budgets $b \in [1, 6]$ bits/value are sampled every 0.001 bits and augmented with retained feasible-frontier breakpoints. The upper row shows the mean dual lower bound and best retained feasible distortion, and the lower row shows the five seed-wise gain upper bounds and their pointwise median.

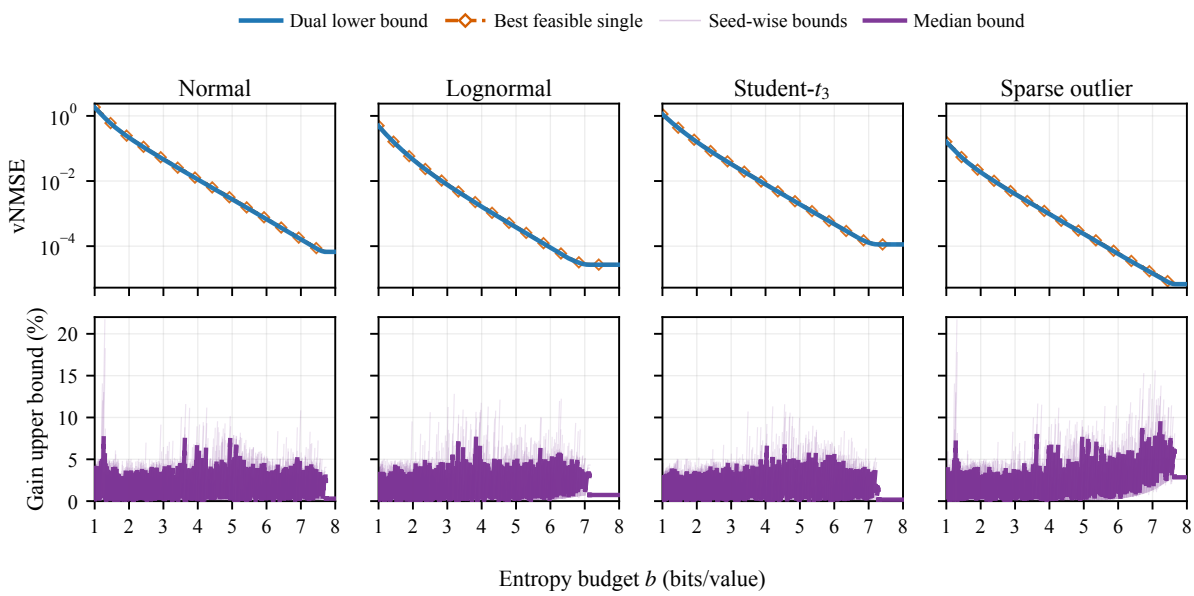


Figure 18: Entropy-constrained vNMSE and the time-sharing gain upper bound with at most $s = 256$ quantization values. The setup and curve encodings match Figure 16: columns show the four BF16 synthetic distributions with $d = 16{,}384$, $P = P_X$; budgets $b \in [1, 8]$ bits/value are sampled every 0.001 bits and augmented with retained feasible-frontier breakpoints. The upper row shows the mean dual lower bound and best retained feasible distortion, and the lower row shows the five seed-wise gain upper bounds and their pointwise median.