High-Rate Nested-Lattice Quantized Matrix Multiplication with Small Lookup Tables

Iris Kaplan
Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
Email: iris.kaplan1@mail.huji.ac.il

Or Ordentlich
Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
Email: or.ordentlich@mail.huji.ac.il

***Abstract*—Recent work have shown that the quantization for matrix multiplication problem can be optimally solved by quantizing each column in each matrix using a nested lattice code, and then multiplying the de-quantized matrices. It was further demonstrated that when product codes of sub-dimension d and rate R are used, the de-quantization and inner product operations can be implemented with querying a lookup table (LUT) of size 2^{2dR}, but this is only useful when dR is sufficiently small. This in turn limits LUT-based inner product decoding to low-rate quantizers. In this work, we develop a rate R hierarchical nested lattice quantization framework, which quantizes each vector to M layers, and admits LUT-based inner product decoding using an LUT of size $2^{2d\frac{R}{M}}$, allowing for high-rate quantization. We provide analytic bounds on the loss of the developed scheme compared to standard nested lattice quantizers, and also numerically illustrate that this loss is negligible. Thus, our scheme enables to use small LUTs without compromising the overall distortion.**

I. Introduction

Matrix multiplication constitutes the major part of the workload associated with inference in deep neural nets (DNNs) and large language models (LLMs). In order to compute $A^\top B$ for two matrices $A \in \mathbb{R}^{n\times a}$ and $B \in \mathbb{R}^{n\times b}$ and store the result, one needs to fetch/store $n(a+b)+ab$ entries from/to memory, whereas the computation requires $2nab$ operations. Modern hardware has become so efficient in performing multiplications and additions, that it is often the memory bandwidth that forms the bottleneck in matrix multiplication, especially when a or b are small. To remedy the limitations posed by the memory bandwidth, much research in the AI community over the last decade was dedicated to reducing the IO burden by compressing/quantizing the entries of one or both matrices A, B [1]–[11]. See [12, Section I.A] for a discussion on what quantization rates are required for fully utilizing the compute cores. As it turns out, LLMs in the generation phase are typically memory-limited, and any reduction in the quantization rate speeds up the inference time proportionally. While on modern GPUs the memory limitation is so severe, that one can afford to spend a few cycles on de-quantization without affecting inference time, in modern CPUs the memory limitation is less acute, and often de-quantization must be highly efficient in order to result in shorter inference time.

Motivated by the above, this work develops a quantization scheme for matrix multiplication with fast decoding, which relies on using lookup tables (LUT) rather than first de-quantizing the elements of A, B and computing their product. In particular, we rely on the nested-lattice based quantization for matrix multiplication scheme from [12], which was shown to be worst-case optimal when high-dimensional quantizers are used. A low-complexity variant of that scheme, based on the product of Voronoi codes [13] in $\mathbb{R}^d$, where d is small, say $3 \le d \le 8$, was further developed in [12], and was numerically shown to perform quite close to the fundamental limit, and attain state-of-the-art results for quantized LLMs [14]. While the encoding in this scheme requires to perform nearest neighbor decoding to the base lattice $L \subset \mathbb{R}^d$, it was pointed out that decoding can be performed via repeated access to an LUT with 2^{2dR} entries, that stores all possible inner products between two vectors in the rate-R Voronoi code. Decoding using LUTs is highly appealing (at least on a CPU), but only if the LUT is small enough to be stored in the fastest cache (L1 cache), which poses a significant constraint on the dimension-rate product dR (say, $dR \le 8-9$ for modern CPUs).

The goal of this work is to circumvent this limitation, and allow for nested-lattice quantization for matrix multiplication which admits efficient LUT-based decoding even for high quantization rate R. To this end, we develop an hierarchical nested lattice quantization scheme. Our scheme quantizes each vector in $\mathbb{R}^d$ to $M \ge 1$ layers, and decoding of the inner product between two quantized vectors in $\mathbb{R}^d$ only requires M^2 queries to a single LUT with $2^{2d\frac{R}{M}}$ entries. We show that despite the reduction of the required LUT size, our scheme's performance is extremely close to that of a Voronoi code with the same rate.

A Python implementation for quantized matrix multiplication using our scheme is available in [15], whereas an efficient C implementation is available in [16].

Related Work: Our construction falls within the framework of successive refinement [17], which is also often referred to as *embedded codes* or *residual vector quantization* in the literature. The quadratic Gaussian rate-distortion problem is known to be successively refinable, that is, successive refinement codes can achieve the optimal rate-distortion tradeoff. Several image compression algorithms, such as Embedded Zerotree Wavelet (EZW) [18] and Set Partitioning in Hierarchical Trees (SPIHT) [19], include a successive refinement component where first the MSBs of certain coefficients are sent, and the LSBs are sent afterwards. In an attempt to extend the MSB-to-LSB "scalar-quantization" refinement used by these algorithms to lattice-based vector quantization, [20] and [21] constructed lattice-based successive refinement schemes in the same spirit as ours. However, [21] only considered layers with rate $R = 1$ bit, whereas the encoding/decoding in [20] did not fully exploit the algebraic structure of nested lattice codes. Furthermore, the encoding in [20], [21] is top-bottom, as is usually the case for successive refinement: one first quantizes the source to the coarsest lattice, and then uses a finer lattice to quantize the residual quantization error, and so on. Our scheme, on the other hand, quantizes the source to the finest lattice, and describes the obtained point as a coset of a coarse lattice whose index grows with the number of layers. As a consequence, we are able to obtain rigorous bounds on the performance of our hierarchical scheme compared to a single-layer Voronoi code of the same rate.

Using a Cartesian product of low-dimensional quantizers for quantizing a high-dimensional vector is a standard practice from the first days of digital communication [22], [23, Chapter 12.7]. More recently, product codes for fast computation of approximate inner product or Euclidean distance were heavily studied within the context of approximate nearest neighbor (ANN) search and information retrieval. The idea of partitioning vectors to small chunks, building a quantizer for each chunk, and constructing corresponding LUTs for fast inner product computations was developed in [24]. The work [24] inspired a huge body of follow-up work. Among the many extensions studied, the combination of product codes with additive quantization seems to be closest to the approach we take in this paper. In additive quantization [25] one constructs the quantizer $Q : \mathbb{R}^d \to [\prod_{m=1}^{M} K_m]$ as the Minkowski sum $\mathcal{C}_1 + \cdots + \mathcal{C}_M$ of M codebooks $\mathcal{C}_1, \ldots, \mathcal{C}_M$ of sizes $K_1, \ldots, K_M$, respectively. Typically, those codebooks are learned from the data using variants of Loyd's algorithm/K-means, and are therefore unstructured, which makes the encoding/decoding quite challenging. Moreover, in general product+additive quantization requires storing many different codebooks and LUTs. Our scheme uses the same lattice for all chucks and all sub-codebooks, and a *single* lookup table for all operations involved in the inner product computation. For codebooks learned via K-means, the entries of the LUTs must be quantized as well, which may degrade performance [26]. In contrast, for standard choices of the base lattice, e.g., the D_n or E_n families, the possible inner products will further be integer-valued, so each entry of the LUT can be efficiently stored. Product codes with LUT based inner product decoding for machine learning applications was considered in [27].

II. Hierarchical Nested-Lattice Quantizers

We first review some basic lattice definitions. See [28] for a comprehensive treatment of lattices in information theory. For a lattice $L \subset \mathbb{R}^d$ we define the nearest neighbor quantizer $Q_L : \mathbb{R}^d \to L$ as

$$Q_L(x) = \underset{\lambda \in L}{\operatorname{argmin}} \|x - \lambda\|, \quad (1)$$

where ties are broken arbitrarily, but in a systematic manner. The Voronoi region $\mathcal{V}_L$ is defined as the set of all points in $\mathbb{R}^d$ that are closer to 0 than to any other lattice point

$$\mathcal{V} = \mathcal{V}_L = \left\{x \in \mathbb{R}^d \ : \ Q_L(x) = 0\right\}. \quad (2)$$

Any lattice $L \subset \mathbb{R}^d$ has a (non-unique) generating matrix $G \in \mathbb{R}^{d \times d}$ such that $L = G\mathbb{Z}^d$. Let $Z \sim \mathrm{Uniform}(\mathcal{V}_L)$ be a random vector uniformly distributed over the Voronoi region of L. We define the second moment of the lattice L as $\sigma^2(L) = \frac{1}{d}\mathbb{E}\|Z\|^2$. For any natural number r we have that $rL \subset L$ and we construct the nested lattice quantizer/Voronoi code [13]

$$\mathcal{A}_r = L \cap (r\mathcal{V}), \quad (3)$$

which is equivalent to the the quotient group $L/rL \cong (\mathbb{Z}/r\mathbb{Z})^d$. This algebraic structure enables to use Voronoi codes as quantizers with fast encoding and decoding schemes. See [13] and also [12, Algorithms 1 and 2]. In particular, these algorithms encode a vector $x \in \mathbb{R}^d$ to $d \log_2 r$ bits, and based on those bits, the decoder outputs $\hat{x} \in \mathcal{A}_r$, where $\hat{x} = Q_L(x)$ whenever $Q_L(x) \in r\mathcal{V}$. The event $\hat{x} \neq Q_L(x)$ is called an *overload* event. The quantization rate of this scheme is $R = \log_2(r)$.

Note that for Voronoi codes, the tie-breaking in (1), which affects the boundary of $\mathcal{V}$ defined in (2), is highly important, since for an even integer r we will always have points of L on the boundary of $r\mathcal{V}$. To circumvent the numerical difficulties associated with treating the boundary, one can fix some very small and unstructured[1] perturbation vector $\varepsilon \in \mathbb{R}^d$ and replace the nearest neighbor quantizer $Q_L(x)$ with $Q_L(x+\varepsilon)$ everywhere.

Our end goal in this paper is to compute inner products based on Voronoi codes and LUTs. To that end we can pre-compute all inner products of pairs of points $\lambda_i, \lambda_j \in \mathcal{A}_r$ and store them in an LUT of size $r^{2d}B = 2^{2dR}B$-bytes, assuming we use B bytes to represent the value of each inner product. In order to allow fast access to the LUT, it must be small enough to fit in the L1 cache. This constrains the product dR, and typical numbers (depending on the processing unit that is used) are $dR \leq 8 - 9$. Our goal is to facilitate the use of Voronoi codes with LUT-based inner product decoding, while allowing arbitrarily large quantization rate R. This will be enabled via hierarchical nested-lattice quantizers.

A. Proposed Hierarchical Nested-Lattice Quantizers

Algorithm 1 Hierarchical Nested-Lattice Encoder

```
**Inputs:** $x \in \mathbb{R}^d$, Lattice $L \subset \mathbb{R}^d$ with generating matrix $G \in \mathbb{R}^{d\times d}$, nesting ratio $q \in \mathbb{N}$, hierarchy depth $M \in \mathbb{N}$
**Outputs:** Encoding vectors $b_0, b_1, \ldots, b_{M-1} \in [q]^d$
$\tilde{g} \leftarrow x$
**for** $m = 0$ to $M - 1$ **do**
    $\tilde{g} \leftarrow Q_L(\tilde{g})$
    $b_m \leftarrow [G^{-1} \cdot \tilde{g}] \mod q$
    $\tilde{g} \leftarrow \tilde{g}/q$
**end for**
$\mathrm{OverloadError} = \mathbb{1}\{Q_L(\tilde{g}) \neq 0\}$
**return** $b_0, b_1, \ldots, b_{M-1}$
```

For a lattice $L \subset \mathbb{R}^d$ and a natural number q, we denote by $Q_{qL}(\cdot)$ the nearest neighbor quantizer for the lattice qL. Note

[1]In particular, we would like to avoid the situation where ε is contained in one of the hyperplanes defining the boundary of $\mathcal{V}$.

Algorithm 2 Hierarchical Nested Lattice Decoder

```
**Inputs:** Encoding vectors $b_0, b_1, \ldots, b_{M-1} \in [q]^d$, Lattice $L \subset \mathbb{R}^d$ with generating matrix $G \in \mathbb{R}^{d\times d}$, nesting ratio $q \in \mathbb{N}$, hierarchy depth $M \in \mathbb{N}$
**Output:** Reconstructed vector $\hat{x} \in L$
$\hat{x} \leftarrow 0$
**for** $m = 0, \ldots, M-1$ **do**
    $x_m \leftarrow G \cdot b_m - q \cdot Q_L((G \cdot b_m)/q)$
    $\hat{x} \leftarrow \hat{x} + q^m x_m$
**end for**
**return** $\hat{x}$
```

that $Q_{qL}(x) = q \cdot Q_L(x/q)$. For a non-negative integer m and $x \in \mathbb{R}^d$, define

$$\begin{aligned} Q^{\circ m}(x) = Q^{\circ m}_{L,q}(x) &= \left(Q_{q^m L} \circ Q_{q^{m-1}L} \cdots \circ Q_L\right)(x) \\ &= Q_{q^m L}\left(Q_{q^{m-1}L}\left(\cdots\left(Q_L(x)\right)\right)\right). \end{aligned} \quad (4)$$

Note that $Q^{\circ m}(x)$ is not equal to $Q_{q^m L}(x)$ in general, unless L, q satisfy the perfect tiling condition: $(L \cap q\mathcal{V}) + \mathcal{V} = q\mathcal{V}$. The perfect tiling condition is met by $L = \mathbb{Z}$ with odd q, but is rarely met by any lattice in dimensions $d > 1$.

Fix a lattice L, and two natural numbers q, M, and recall that $\mathcal{A}_q = L \cap (q\mathcal{V})$. Our scheme encodes each $x \in \mathbb{R}^d$ to $d{\cdot}M \log_2(q)$ bits, and based on those bits the decoder outputs a point $\hat{x}$ in the constellation $\mathcal{C}_{L,q,M} = \sum_{m=0}^{M-1} q^m \mathcal{A}_q$ where the sum above is a Minkowski sum. Whenever $Q_L(x) \in \mathcal{C}_{L,q,M}$ we have that the reconstruction produced by our scheme satisfies $\hat{x} = Q_L(x)$. Figure 2 depicts the constellation $\mathcal{C}_{L,q,M}$ for the hexagonal lattice $L = A_2$, $q = 6$ and $M = 3$.

To describe our scheme, define $\tilde{g}_m(x) = \frac{Q^{\circ m}(x)}{q^m}$, and note that $\tilde{g}_m \in L$ by definition, and can be computed recursively as

$$\tilde{g}_0(x) = Q_L(x); \quad \tilde{g}_m(x) = Q_L\left(\frac{\tilde{g}_{m-1}(x)}{q}\right), \quad m = 1, \ldots.$$

For $m = 0, \ldots, M-1$, let

$$\begin{aligned} g_m(x) &= Q^{\circ m}(x) - Q^{\circ(m+1)}(x) \\ &= Q^{\circ m}(x) - Q_{q^{m+1}L}\left(Q^{\circ m}(x)\right) \\ &= q^m\left(\frac{Q^{\circ m}(x)}{q^m} - Q_{qL}\left(\frac{Q^{\circ m}(x)}{q^m}\right)\right) \quad (5) \\ &= q^m\left(\tilde{g}_m(x) - Q_{qL}\left(\tilde{g}_m(x)\right)\right). \quad (6) \end{aligned}$$

Since $\tilde{g}_m(x) \in L$, we clearly have that $g_m(x) \in q^m(L \cap q\mathcal{V}) = q^m \mathcal{A}_q$. Furthermore, we can use the standard Voronoi encoder/decoder [13], [12, Algorithms 1 and 2] for representing $\tilde{g}_m(x) - Q_{qL}(\tilde{g}_m(x)) \in \mathcal{A}_q$ using $d \log_2(q)$ bits, and mapping those bits back to $\mathcal{A}_q$. Algorithm 1 implements the encoder in our scheme. It recursively computes $\{\tilde{g}_m\}_{m=0}^{M-1}$ and represents each of them using the sequences $\{b_m\}_{m=0}^{M-1}$, each of length $d \log_2(q)$ bits. The corresponding decoder is given in Algorithm 2. It recovers from each vector b_m the corresponding point $\tilde{g}_m(x) - Q_{qL}(\tilde{g}_m(x))$ in $\mathcal{A}_q$, and outputs the reconstruction

$$\hat{x} = \sum_{m=0}^{M-1} q^m\left[\tilde{g}_m(x) - Q_{qL}\left(\tilde{g}_m(x)\right)\right] = \sum_{m=0}^{M-1} g_m(x). \quad (7)$$

Lemma 1: Let $x \in \mathbb{R}^d$ and let $\hat{x} \in L$ be its reconstruction using the hierarchical nested-lattice quantizer, that is the output of Algorithm 2 applied on the output of Algorithm 1. Then $\hat{x} = Q_L(x)$ iff $Q^{\circ M}(x) = 0$.

Proof. Telescoping $\sum_{m=0}^{M-1} g_m(x)$ in (7), we observe that

$$\hat{x} = Q^{\circ 0}(x) - Q^{\circ M}(x) = Q_L(x) - Q^{\circ M}(x). \quad (8)$$

Thus, $\hat{x} = Q_L(x)$ iff $Q^{\circ M}(x) = 0$. ■

As a consequence of Lemma 1, we see that the binary variable OverloadError computed in Algorithm 1 indeed satisfies $\text{OverloadError} = \mathbb{1}\{\hat{x} \neq Q_L(x)\}$.

Scaling and dithering: In order to get the smallest distortion using the proposed scheme, as well as when using standard Voronoi codes, one scales the constellation by a factor $\beta > 0$. The granular error is $\sigma^2(\beta L) = \beta^2\sigma^2(L)$ and is increasing in β. On the other hand, the overload probability is decreasing in β. Thus, one typically looks for the smallest β for which the overload probability is "small enough". Often, one also uses a *dither* vector $z \in \mathcal{V}$ to shift the constellation. Specifically, to implement scaling by $\beta > 0$ and dithering by $z \in \mathcal{V}$, we set the input to our encoder as $\frac{x}{\beta} - z$ instead of x, and the output of our decoder as $\beta(\hat{x} + z)$ instead of $\hat{x}$.

Overload avoidance mechanism: We employ an overload avoidance mechanism similar to the one introduced in [12]. Specifically, we first set $\beta = \beta_0$ and $T = 0$. We input $\frac{x}{\beta} - z$ to our encoder, and check whether $\text{OverloadError} = 0$. If so, we send the encoded bits $b_0, \ldots, b_{M-1}$. Otherwise, we set $\beta \leftarrow 2^\alpha \beta$, $T \leftarrow T+1$, where $\alpha > 0$ is a parameter of the algorithm, and try again, and so on until $\text{OverloadError} = 0$ and we send the encoded bits $b_0, \ldots, b_{M-1}$. We also send an entropy-coded description of T to the decoder, using $\approx H(T)$ bits. In total, the expected rate of this scheme is $M \log_2(q) + \frac{H(T)}{d}$. The decoder, in turn, reconstructs T from the entropy coded bits and $\hat{x}$ from $b_0, \ldots, b_{M-1}$, and outputs $2^{\alpha T}\beta_0(\hat{x} + z)$.

Successive Refinement: In some situations one may wish to start by recovering the source x with low resolution and gradually improve the resolution as more bits describing it become available. If we run the for-loop in Algorithm 2 from $m = M-1$ down to $m = 0$, we will get a gradually improving description. Moreover, if we only recover $t < M$ layers, by running that for-loop from $i = M-1$ down to $i = M-t$, we will recover $Q^{\circ(M-t)}(x) - Q^{\circ M}(x)$, which equals $Q^{\circ(M-t)}(x)$ if overload did not occur.

B. Bounds and numerical results

Let $\mathcal{P}_{q,M} = \{x \in \mathbb{R}^d : Q^{\circ M}(x) = 0\}$, and note that $\mathcal{P}_{q,M}$ is a fundamental cell of the lattice $q^M L$. Note that furthermore $\mathcal{C}_{L,q,M} = L \cap \mathcal{P}_{q,M}$, and it therefore follows that $\mathcal{C}_{L,q,M} \cong L/q^M L$. A Voronoi code $\mathcal{A}_{q^M}$ selects the coset representatives with the lowest energy, and consequently approximately minimizes the overload probability among all choices of coset representatives of $L/q^M L$. While the hierarchical scheme described in the previous subsection has several advantages over a Voronoi code of the same rate and same lattice L, it selects the coset representatives of $L/q^M L$ as $L \cap \mathcal{P}_{q,M}$, and is therefore inferior to the corresponding Voronoi code in terms of overload probability. The following result shows that

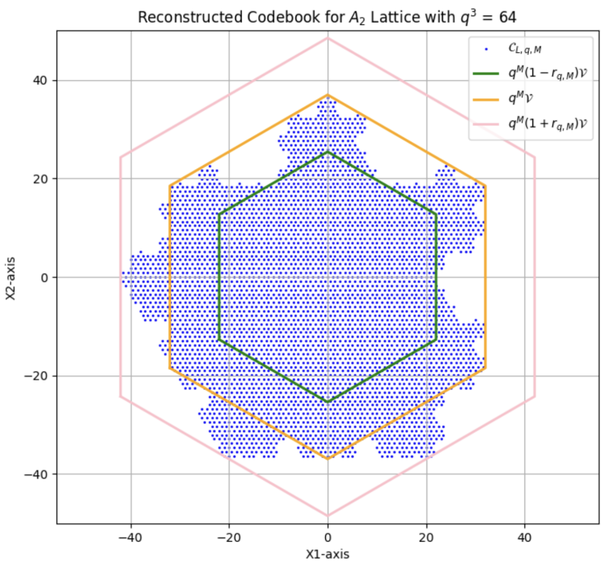


Fig. 1: Codebook of the hierarchical nested lattice quantizer with $L = A_2$, $q = 4$ and $M = 3$.

the overload probability of $\mathcal{C}_{L,q,M}$ is nevertheless upper (resp. lower) bounded by the overload probability of a Voronoi code whose rate is $\log\left(1 \mp \frac{1-q^{1-M}}{q-1}\right)$ bits smaller (resp. greater).

Lemma 2:

$$\mathcal{A}_{q^M(1-r_{q,M})} \subset \mathcal{C}_{L,q,M} \subset \mathcal{A}_{q^M(1+r_{q,M})}, \tag{9}$$

where $r_{q,M} = q^{-M}\sum_{m=1}^{M-1} q^m = \frac{1-q^{1-M}}{q-1}$.

The green and pink scaled Voronoi regions in Figure 2 illustrate the inclusion in (9).

Proof. Recall that since $\mathcal{V}$ is a convex set in $\mathbb{R}^d$, we have that $\alpha\mathcal{V} + \beta\mathcal{V} = (\alpha+\beta)\mathcal{V}$ for all $\alpha, \beta > 0$. To prove that $\mathcal{C}_{L,q,M} \subset \mathcal{A}_{q^M(1+r_{q,M})}$, note that

$$\begin{aligned}\mathcal{C}_{L,q,M} &= \sum_{m=0}^{M-1} q^m(L \cap q\mathcal{V}) \subset L \cap \left(q\mathcal{V} \cdot \sum_{m=0}^{M-1} q^m\right) \\ &= L \cap ((q^M + \sum_{m=1}^{M-1} q^m)\mathcal{V}).\end{aligned} \tag{10}$$

To prove $\mathcal{A}_{q^M(1-r_{q,M})} \subset \mathcal{C}_{L,q,M}$, it suffices to show that for any $y \in \mathcal{A}_{q^M(1-r_{q,M})}$ it holds that $Q^{\circ M}(y) = 0$. We have,

$$\begin{aligned}&Q^{\circ(M-1)}(y) = Q_{q^{M-1}L}\left(Q^{\circ(M-2)}(y)\right) \\ &\in Q^{\circ(M-2)}(y) + q^{M-1}\mathcal{V} \in \cdots \in Q_L(y) + \sum_{m=1}^{M-1} q^m\mathcal{V}\end{aligned} \tag{11}$$

and since $y \in \mathcal{A}_{q^M(1-r_{q,M})}$, we also have $y = Q_L(y) \subset q^M(1-r_{q,M})\mathcal{V}$. Consequently,

$$Q^{\circ(M-1)}(y) \in q^M(1-r_{q,M})\mathcal{V} + \sum_{m=1}^{M-1} q^m\mathcal{V} = q^M\mathcal{V}, \tag{12}$$

which implies that $Q^{\circ M}(y) = Q_{q^M L}\left(Q^{\circ(M-1)}(y)\right) = 0$ ■

Simulation results: We plot the distortion-rate tradeoff attained by three different nested lattice quantization schemes: (a) Voronoi code with $r = q^M$ (b)Voronoi code with $r = q^M(1-r_{q,M})$ (c)The developed hierarchical scheme with M layers and nesting ratio q. For all schemes we use the same base lattice L, and use the overload avoidance mechanism described above, with $\alpha = 1/3$ and β_0 optimized separately for each of the three schemes. In the simulations we use $L = D_4$, $M = 2$, and $q \in \{3, 4, \ldots, 9\}$. We quantize $N = 5000$ iid realizations of a $\mathcal{N}(0, I_4)$ source, and plot the obtained distortion-rate tradeoff for each scheme in Figure 2a. We also plot the Shannon limit $D(R) = 2^{-2R}$ for reference. It can be seen that the hierarchical nested-lattice quantizer is indeed strictly better than the (lower-rate) Voronoi code with $r = q^M(1-r_{q,M}) = q(q-1)$, and is almost as good as the reference Voronoi code with $r = q^M = q^2$. Remarkably, using the overload avoidance mechanism, both schemes are less than $1/2$ bit away from the fundamental limit, using a simple code of dimension $d = 4$.

III. Fast Inner-Product Computation

Our main motivation for introducing the hierarchical scheme was to develop a fast decoder for the quantization for inner product computation problem. Let x and y be two vectors in $\mathbb{R}^d$ that were quantized by the hierarchical scheme above. We have that

$$\hat{x} = \sum_{i=0}^{M-1} q^i \hat{x}_i; \quad \hat{y} = \sum_{j=0}^{M-1} q^j \hat{y}_j, \tag{13}$$

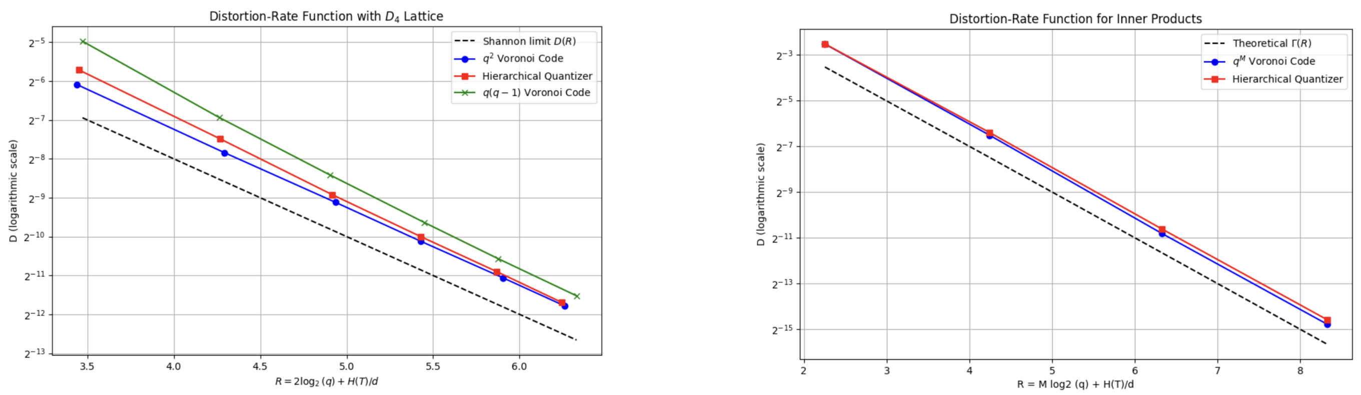
where $\hat{x}_i, \hat{y}_j \in \mathcal{A}_q$ for all $i, j \in \{0, \ldots, M-1\}$. Consequently,

$$\hat{x}^\top \hat{y} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} q^{i+j} \cdot \hat{x}_i^\top \hat{y}_j. \tag{14}$$

Note further that all inner products $\hat{x}_i^\top \hat{y}_j$ participating in the sum above, involve two vectors in $\mathcal{A}_q$. Furthermore, the vector $\hat{x}_i$ is represented by the encoding vector $b_i(x) \in [q]^d$ in Algorithm 1 (in particular, we have $\hat{x}_i = G \cdot b_i(x) - q \cdot Q_L(G \cdot b_i(x))/q)$), and similarly $\hat{y}_j$ is represented by the encoding vector $b_j(y) \in [q]^d$. We can therefore pre-compute all q^{2d} inner products in $\mathcal{A}_q \times \mathcal{A}_q$ and store the results in an LUT $\{\mathcal{L}(b_i, b_j)\}_{b_i, b_j \in [q]^d \times [q]^d}$ indexed by two encoding vectors b_i, b_j. It therefore follows that for if our goal is to quantize $x, y \in \mathbb{R}^d$ in order to compute an approximation for $x^\top y$, we can quantize each of them to $b_0(x), \ldots, b_{M-1}(x)$ and $b_0(y), \ldots, b_{M-1}(y)$ using Algorithm 1 with rate $R = M\log_2(q)$ bits per entry, and then decode the inner product via

$$\hat{x}^\top \hat{y} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} q^{i+j} \cdot \mathcal{L}(b_i(x), b_j(y)). \tag{15}$$

The complexity of decoding the inner product $\hat{x}^\top \hat{y}$ therefore consists of querying the LUT M^2 times, M^2 scalar multiplications of the fetched LUT values by q^{i+j} (if q is a power of 2 this can be implemented by simple bit-shifts) and M^2 additions. As mentioned above, the main gain of the hierarchical scheme is that we only need to store a single LUT



(a) Vector quantization, $M = 2$, q varies

(b) Inner product quantization, $q = 4$, M varies

Fig. 2: Distortion-Rate curves for nested lattice quantizers, $L = D_4$.

of size $2^{d\log_2(q)} = 2^{2d\frac{R}{M}}$, instead of an LUT of size 2^{2dR} that is needed if standard Voronoi codes are used. The "price" for the reduction of the LUT size is that we need to access it M^2 times.

Scaling and dithering: As in the previous section, one can scale and shift the constellation by encoding $\frac{x}{\beta_1} - z_1$ rather than x and $\frac{y}{\beta_2} - z_2$ rather than y, where $\beta_1, \beta_2 > 0$ and $z_1, z_2 \in \mathcal{V}$. The decoder in turn, should output $\beta_1\beta_2(\hat{x} + z_1)^\top(\hat{y} + z_2)$ rather than $\hat{x}^\top\hat{y}$. Taking arbitrary dither vectors $z_1, z_2 \in \mathcal{V}$ does not permit to decode $(\hat{x}+z_1)^\top(\hat{y}+z_2)$ only by accessing the LUT $\mathcal{L}$, as in (15). To circumvent this issue, we restrict the dither vectors to the constellation $q^{-1}\mathcal{A}_q \subset \mathcal{V}$. In particular, we choose two vectors $b_{z_k} \in [q]^d$, $k = 1, 2$ and set $z_k = q^{-1}\left[G \cdot b_{z_k} - q \cdot Q_L((G \cdot b_{z_k})/q)\right]$ as our dither vectors. We can then define $b_{-1}(x) = b_{z_1}$ and $b_{-1}(y) = b_{z_2}$ and compute

$$(\hat{x} + z_1)^\top(\hat{y} + z_2) = \sum_{i=-1}^{M-1}\sum_{j=-1}^{M-1} q^{i+j} \cdot \mathcal{L}(b_i(x), b_j(y)). \quad (16)$$

One-sided quantization: There are many practical scenarios where one needs to compute many inner products $y^\top x_k$, $k = 1, \dots, K$ between a fixed vector $y \in \mathbb{R}^d$ and many other vectors $x_1, \dots, x_K \in \mathbb{R}^d$, $k \gg 1$. For instance, $x_1, \dots, x_K$ can be vectors in a database, and y is a query for which one needs to find the approximate nearest neighbor in the database. In such situations, it is often the case that y can be stored essentially in full resolution, but the vectors in the database must be quantized. Assuming we quantize each x_k (assuming no dithering for simplicity) using our hierarchical scheme, we can compute each inner product as

$$y^\top \hat{x}_k = \sum_{i=0}^{M-1} q^i \mathcal{L}_y(b_i(x_k)), \quad k = 1, \dots, K, \quad (17)$$

where $\mathcal{L}_y$ is an LUT of size $q^d = 2^{d\frac{R}{M}}$ consisting of the values of all inner products between y and a vector in $\mathcal{A}_q$.

Arbitrary dimension via product codes: In order to solve the quantization for inner product computation problem for vectors $x, y \in \mathbb{R}^n$, $n \gg 1$, we use a product of quantizers for $\mathbb{R}^d$. Let $K = n/d$, and assume for simplicity that K is an integer. We may split x and y to K chunks, each of size d, such that $x = [x_1^\top | \cdots | x_K^\top]^\top$, $y = [y_1^\top | \cdots | y_K^\top]^\top$. We can then quantize each chunk x_k (resp. y_k) to $\hat{x}_k$ (resp. $\hat{y}_k$) using our hierarchical d-dimensional scheme, and decode the inner product as

$$\hat{x}^\top\hat{y} = \sum_{k=1}^{K} \hat{x}_k^\top \hat{y}_k. \quad (18)$$

Universality via random rotation: Our hierarchical quantizer is tailored to a Gaussian iid source, and may not perform well when the source to be quantized does not resemble a Gaussian vector. In order to obtain a universal quantization for inner product scheme, whose performance depends only on $\|x\|_2$, $\|y\|_2$ and $|x^\top y|$, one should draw a random rotation matrix $S \in \mathbb{R}^{n\times n}$, and quantize $Sx/\|x\|_2$ (resp. $Sy/\|y\|_2$) instead of x (reps. y). The inner product between the quantized vectors should then be scaled back by $\|x\|_2\|y\|_2$. See [12] for details and and analysis.

Simulation results: We draw $N = 5000$ pairs of iid vectors $X, Y \sim \mathcal{N}(0, I_n)$ where $n = 512$. We use $d = 4$. Hence, for each pair, we quantize each vector by chunking it to $K = 512/4 = 128$ pieces, quantize each piece using a lattice quantizer based on $L = D_4 \subset \mathbb{R}^4$ with (a)Voronoi codes with $r = q^M$ where $q = 4$ and M varies (b)Hierarchical nested-lattice quantizer with $q = 4$ and varying M, and computing the inner product $\hat{X}^\top\hat{Y}$ as in (18). For the hierarchical scheme, we use the LUT approach (15). For both schemes we use the overlaod avoidance mechanism described above, with $\alpha = 1/3$. We define the distortion as $D = \frac{1}{n}\mathbb{E}(X^\top Y - \hat{X}^\top\hat{Y})^2$, and also plot the fundamental limit $D \geq \Gamma(R)$ from [12] (for $R > 0.906$ we have $\Gamma(R) = 2 \cdot 2^{-2R} - 2^{-4R}$). The results are plotted in Figure 2b for $M = 1, 2, 3, 4$. It is evident that the hierarchical scheme has performance very close to that of Voronoi codes with the same rate, and that both schemes are about half a bit away of the fundamental limit.

Acknowledgment

This work was supported by the ISRAEL SCIENCE FOUNDATION (ISF), grant No.1641/21.

References

[1] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[2] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[3] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[4] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 168–27 183, 2022.

[5] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.

[6] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei, "The era of 1-bit llms: All large language models are in 1.58 bits," *arXiv preprint arXiv:2402.17764*, 2024.

[7] A. Tseng, J. Chee, Q. Sun, V. Kuleshov, and C. De Sa, "Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks," *arXiv preprint arXiv:2402.04396*, 2024.

[8] A. Tseng, Q. Sun, D. Hou, and C. De Sa, "Qtip: Quantization with trellises and incoherence processing," *arXiv preprint arXiv:2406.11235*, 2024.

[9] S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman, "Quarot: Outlier-free 4-bit inference in rotated llms," *arXiv preprint arXiv:2404.00456*, 2024.

[10] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," *Advances in neural information processing systems*, vol. 29, 2016.

[11] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "OPTQ: Accurate quantization for generative pre-trained transformers," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=tcbBPnfwxS

[12] O. Ordentlich and Y. Polyanskiy, "Optimal quantization for matrix multiplication," *arXiv preprint arXiv:2410.13780*, 2024.

[13] J. Conway and N. Sloane, "A fast encoding method for lattice codes and quantizers," *IEEE Transactions on Information Theory*, vol. 29, no. 6, pp. 820–824, 1983.

[14] S. Savkin, E. Porat, O. Ordentlich, and Y. Polyanskiy, "NestQuant: Nested lattice quantization for matrix products and LLMs," *arXiv preprint arXiv:2502.09720*, 2025.

[15] I. Kaplan, "A python package for neseted-lattice LUT," 2025. [Online]. Available: https://github.com/iriskaplan/LatticeQuant

[16] O. Meir, "A C package for neseted-lattice LUT," 2025. [Online]. Available: https://github.com/orimeirgit/NestedLatticeLut

[17] W. H. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Transactions on information theory*, vol. 37, no. 2, pp. 269–275, 1991.

[18] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3445–3462, 1993.

[19] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on circuits and systems for video technology*, vol. 6, no. 3, pp. 243–250, 1996.

[20] D. Mukherjee and S. K. Mitra, "Successive refinement lattice vector quantization," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1337–1348, 2002.

[21] G. Fuchs, "Embedded voronoi codes for successive refinement lattice vector quantization," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 5805–5809.

[22] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[23] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Springer Science & Business Media, 2012, vol. 159.

[24] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[25] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938.

[26] D. W. Blalock and J. V. Guttag, "Bolt: Accelerated data mining with fast vector compression," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 727–735.

[27] D. Blalock and J. Guttag, "Multiplying matrices without multiplying," in *International Conference on Machine Learning*. PMLR, 2021, pp. 992–1004.

[28] R. Zamir, *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory*. Cambridge University Press, 2014.