Optimal Quantization for Matrix Multiplication

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

Yury Polyanskiy
Dept. EECS
MIT
Cambridge, MA, USA
Email: yp@mit.edu

***Abstract*—Recent work in machine learning community proposed multiple methods for performing lossy compression (quantization) of large matrices. This quantization is important for accelerating matrix multiplication (main component of large language models), which is often bottlenecked by the speed of loading these matrices from memory. Unlike classical vector quantization and rate-distortion theory, the goal of these new compression algorithms is to be able to approximate not the matrices themselves, but their matrix product. Specifically, given a pair of real matrices A, B an encoder (compressor) is applied to each of them independently producing descriptions with R bits per entry. These representations subsequently are used by the decoder to estimate matrix product $A^\top B$. In this work, we provide a non-asymptotic lower bound on the mean squared error of this approximation (as a function of rate R) for the case of matrices A, B with iid Gaussian entries. Algorithmically, we construct a universal quantizer based on nested lattices with an explicit guarantee of approximation error for any (non-random) pair of matrices A, B in terms of only Frobenius norms $\|\bar{A}\|_F, \|\bar{B}\|_F$ and $\|\bar{A}^\top \bar{B}\|_F$, where $\bar{A}, \bar{B}$ are versions of A, B with zero-centered columns, respectively. For iid Gaussian matrices our quantizer achieves the lower bound and is, thus, asymptotically optimal. In particular, we derive the rate-distortion function for matrix multiplication of iid Gaussian matrices, which exhibits an interesting phase-transition at $R \approx 0.906$ bit/entry. An extended version of this paper is available in [1].**

I. Introduction

Matrix multiplication is a key component of many numerical algorithms, and is often the dominant factor in the runtime of a program. With the surge of deep neural nets (DNNs) and large language models (LLMs), finding more efficient ways to perform matrix multiplication have become one of the most pressing challenges. Classical work in this field focused on minimizing the number of required operations [2]–[5]. Specifics of contemporary problems, however, require rethinking this classical approach to matrix multiplication. First, in machine learning applications requirements for precision of computing matrix products are quite lax. Second, modern computational hardware is often bottlenecked by the memory bandwidth. A natural solution explored by many researchers is to apply lossy compression to matrices leading to deterioration in precision but improvement in the amount of data transferred between memory and computation cores.

We formalize this problem as follows. Consider a pair of matrices $A \in \mathbb{R}^{n\times a}$ and $B \in \mathbb{R}^{n\times b}$ which need to be described using R bits per entry (using separate compressors), such that a decoder that obtains bit descriptions of both matrices can estimate $\widehat{A^\top B}$. The metric for gauging quality of approximation that we will use is the squared error between ab entries of $\widehat{A^\top B}$ and $A^\top B$. Note that unlike classical vector quantization, we are requiring compression algorithms to be tailored to the special task of matrix multiplication. As a practical motivation, in Section I-A below we argue that reducing R down to a few bits/entry is necessary for LLMs to fully leverage modern matrix multiplication hardware.

A. Importance of quantization for modern applications

To set the stage for the problem, let us estimate what level of quantization (in bits / entry) would be relevant for today's main consumer of matrix multiplications: the large language models (LLMs). For those, quantization is typically employed for accelerating inference. During inference LLM is busy computing many products $A^\top B$ of matrices with sizes $d \times a$ and $d \times b$ respectively. This requires $2abd$ FLOPs and $ad + bd + ab$ entries to load/store from memory. Ideally, we would want to quantize entries in such a way that all compute is fully utilized. For that we need to know the ratio ξ of available FLOPs to available memory bandwidth, a quantity known as "ops:bytes" of a processor. It ranges from $\xi = 5\ldots 20$ for modern CPUs (FP32 arithmetic via AVX512) to $\xi \approx 300$ for the fastest GPUs (FP16 on an H100 or B200). The quantization rate saturating compute should then be bounded (in bits/entry) as

$$R < \frac{16}{\xi}\frac{ab}{a+b+\frac{ab}{d}}\,. \tag{1}$$

It turns out that there are two stages of running inference with LLMs: the pre-fill (when the input prompt is processed) and the generation (when response tokens are sequentially generated). During the pre-fill LLM we have $a = d$ and $b = L$ (d is the so-called hidden dimension and L is the sequence length), while during the generation we have $a = L$ and $b = 1$ (the A matrix coming from KV-cache and B matrix being new token's embedding). Thus, to saturate the computation core, we need

$$R_{\text{pre-fill}} = \frac{16Ld}{\xi(d+2L)}\,, \qquad R_{\text{generate}} = \frac{16L}{\xi(L+1+L/d)} \approx \frac{16}{\xi}\,.$$

We can see that during generation phase, on CPUs we would want to approach 1-3 bits/entry, while on GPUs we will not be able to ever saturate compute (that is, a decrease

in quantization rate translates proportionally to decrease in runtime). For the pre-fill phase, for large LLMs we get $R_{\text{pre-fill}} > 16$ bit (that is, just storing plain FP16 is already good enough). Quantization during pre-fill might still be important for "small" LLMs running on fast GPUs: for example, for BERT [6] we have $L = 512$, $d = 768$ and $\xi = 300$ (for an H100), resulting in quantization rate $R \approx 11.7$ bit/entry.

B. Related work

Randomized linear algebra/sketching, and locality-sensitive hashing (LSH) are techniques widely used in practice for computing approximate inner products and approximate matrix multiplications, as well as other operations, in reduced dimensions. The figure of merit in these fields is typically the tradeoff between the reduced dimension and the approximation error, and many algorithms have been developed for addressing different regimes, see e.g. [7]–[12]. Since the dimension of the reduced matrix/vector is related to the number of bits required for storing it, this body of work is relevant to our study. However, the tradeoff between the number of bits per dimension and the total approximation error, and its dependence on the properties of A, B and $A^\top B$ is often subtle. Thus, there is no immediate translation between the required dimension of a sketch and the number of bits needed for representing it for obtaining the required accuracy.

The topic of matrix quantization has received much attention in the last decade in the context of DNNs and LLMs. The goal here is to reduce the memory footprint of the weight matrices, allowing to load them to the main memory using less IOs, as well as speed up the multiplications and additions operations by moving from floating point numbers to small integers (and when possible, also sparsifying the matrices, saving some operations altogether). Roughly speaking, one can distinguish between two paradigms: *quantization-aware training*, where the training procedure is designed to output weight matrices with "cheap" representation [13], [14], and *post-training quantization*, where the training procedure is performed in high precision, and quantization of the weights is only performed after training has terminated (perhaps with some fine tuning afterwards) [15]–[22]. In order to further speed up matrix multiplication, and reduce the number of IOs needed for using KV-cache, some works also develop quantizers for the activations [16], [18]–[20], [23], while other works assume the activations are kept in high precision [15], [21]. Quantization for DNNs and LLMs are typically evaluated according to the end-to-end performance of the quantized architecture, but often the Frobenius norm of the approximation error is considered as the intermediate optimization criterion for quantizing the weights at each layer [13], [24].

To the best of our knowledge, there was very little work on distributed compression for inner product/matrix multiplication in the information theory literature. Recently, Malak [25] studied the problem of *lossless* distributed compression of binary random matrices for computing their product, and derived non-trivial bounds under stringent assumptions on the joint distribution. Some prior work considered the problem of distributed compression of random vectors with the goal of approximately computing a linear function of those vectors [26], [27]. In those works, the goal was to estimate, say, the difference between the two vectors in $\mathbb{R}^n$, which is itself a vector in $\mathbb{R}^n$. While the inner product of these vectors, which is a scalar in $\mathbb{R}$, can be computed from their difference (assuming their individual norms were encoded in high resolution), it seems, a-priory, that distributed compression for inner product computation is an easier task. Our results show that this is, in fact, hardly the case. Another line of related work in the information theory literature, is that of Ingber et al. [28] that considered the fundamental limits of lossy compression of a database in order to support approximate nearest neighbor search (see also [29] for a practical implementation). We note in passing that much recent work focused on coding for speeding up distributed matrix multiplication by introducing redundancy for mitigating the effect of "slow workers" (stragglers), see, e.g., [30]. This line of work is not directly related to approximate matrix multiplication via compression, studied in this paper.

II. Main Results

Our main result shows existence of universal quantizers (based on lattices) which compress A and B to R bits/entry and come with explicit precision guarantees. Furthermore, we also show that these guarantees cannot be generally improved by proving a matching lower bound for the case of matrices A and B with iid Gaussian entries. We emphasize, though, that quantizers *are* universal and do not require Gaussian matrices.

To introduce our main results, let us define the function

$$\Gamma(R) = \begin{cases} 1 - \left(1 - \left(2 \cdot 2^{-2R^*} - 2^{-4R^*}\right)\right) \frac{R}{R^*} & R < R^* \\ 2 \cdot 2^{-2R} - 2^{-4R} & R \geq R^* \end{cases}. \tag{2}$$

where $R^* \approx 0.906$ is the solution to the fixed-point equation

$$R = \frac{1}{2} \log_2(1 + 4R \ln 2) \tag{3}$$

It will turn out that $\Gamma(R)$ is distortion-rate function for the matrix multiplication of iid Gaussian matrices.

We say that a matrix $A \in \mathbb{R}^{n \times m}$ has "M-bounded entries" if $|a_{i,j}| \in \{0\} \cup [M^{-1}, M]$ for all $i \in [n], j \in [m]$. Our results require the matrices A and B to have M-bounded entries, with $M = e^{o(n)}$. To be more concrete, throughout this paper we take $M = n^{10} 2^{2000}$. In particular, this choice of M guarantees that matrices represented in FP64 format have bounded entries. This extremely mild condition guarantees that we can describe the ℓ_2 norm of each column of A, B with small multiplicative error using $o(n)$ bits. Let $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$ be the all-ones vector. For a column vector $x \in \mathbb{R}^n$ we denote by $\bar{x} = x - \left(\frac{1}{n} \mathbf{1}^\top x\right) \mathbf{1}$ its zero-centered version. For a matrix $A = [a_1| \cdots |a_a] \in \mathbb{R}^{n \times a}$ we denote $\bar{A} = [\bar{a}_1| \ldots |\bar{a}_a]$. Our first result is the following.

Theorem 1: For any $\varepsilon > 0$ and sufficiently large n, there exist randomized encoders $f_1 : \mathbb{R}^{n \times a} \to [2^{naR}]$, $f_2 : \mathbb{R}^{n \times b} \to$

$[2^{nbR}]$, and decoder $g : [2^{naR}] \times [2^{nbR}] \to \mathbb{R}^{a\times b}$ such that for any $A \in \mathbb{R}^{n\times a}$ and $B \in \mathbb{R}^{n\times b}$ with bounded entries we have

$$\mathbb{E}\|A^\top B - g(f_1(A), f_2(B))\|_F^2 < \|\bar{A}^\top \bar{B}\|_F^2 \cdot \left(\Gamma^2(R) + \varepsilon\right) + \frac{\|\bar{A}\|_F^2 \|\bar{B}\|_F^2}{n} \left(\Gamma(R) - \Gamma^2(R) + \varepsilon\right) + a \cdot b \cdot n^{-8}.$$

Remark 1: The full statement in [1] of Theorem 1, as well as that of Theorem 3 below, also treats the per-entry MSE distortion, and the "one-sided" case where only A needs to be quantized and B is given in full-resolution.

Our scheme operates by compressing each column of A and B using the same (randomized) nested lattice quantizer $f_{\text{col}} : \mathbb{R}^n \to [2^{nR}]$, which is applied repeatedly to every column, whereas the decoder g simply estimates each column to get matrices $\hat{A}$ and $\hat{B}$ and computes their scaled matrix product; see Figs. 1 and 2. The parameter κ shown in Figures is used by the encoders for time-sharing/sparsification and is set to $\kappa = \min\{R/R^*, 1\}$ in the Theorem. In particular, for $R < R^*$ a fraction $1 - (\frac{R}{R^*})$ of coordinates are ignored (mapped to 0), corresponding to $\kappa = R/R^*$. As it turns out, this dimensionality reduction (à la Johnson-Lindenstrauss) turns out to be necessary to achieve asymptotically optimal distortion.

To get a feel for Theorem 1 let us consider independent matrices A and B drawn iid Gaussian $\mathcal{N}(0, \sigma^2)$. For large n, such matrices have bounded entries and are also arbitrarily close to their centered version, with high probability. We have that $\mathbb{E}\|A^\top B\|_F^2 = \frac{\mathbb{E}\|A\|_F^2 \|B\|_F^2}{n} = \sigma^4 \cdot nab$ in this case and Theorem 1 shows estimate

$$\mathbb{E}[\|A^\top B - \widehat{A^\top B}\|_F^2] \le \sigma^4 nab(\Gamma(R) + \epsilon)\,.$$

It turns out that this is the best possible approximation (at this compression rate), as shown in our next result.

Theorem 2: Let $A \in \mathbb{R}^{n\times a}$ and $B \in \mathbb{R}^{n\times b}$ be independent random matrices, with iid $\mathcal{N}(0, \sigma^2)$ entries. For any $n \ge 1$, and any pair of rate-R encoders $f_1 : \mathbb{R}^{n\times a} \to [2^{naR}]$, $f_2 : \mathbb{R}^{n\times b} \to [2^{nbR}]$ and decoder $g : [2^{naR}] \times [2^{nbR}] \to \mathbb{R}^{a\times b}$, we have

$$\mathbb{E}\|A^\top B - g(f_1(A), f_2(B))\|_F^2 \ge \sigma^4 \cdot nab \cdot \Gamma(R). \quad (4)$$

In other words, the encoders f_1, f_2, g from Theorem 1 attain the lower bound from Theorem 2, and are therefore asymptotically optimal for this class of matrices.

We also show a simpler to use bound, based on our compression scheme applied with no "MMSE scaling" and no time-sharing - that is, with $\alpha = \kappa = 1$ in Figures 1, 2. The resulting bound does not meet the lower bound of Theorem 2 for Gaussian iid matrices. However, for moderate R it is never much worse than the bound from Theorem 1. For some matrices A, B it is significantly better than the bound from Theorem 1.

Theorem 3: For any $\varepsilon > 0$ and sufficiently large n, there exist randomized encoders $f_1 : \mathbb{R}^{n\times a} \to [2^{naR}]$, $f_2 : \mathbb{R}^{n\times b} \to [2^{nbR}]$, and decoder $g : [2^{naR}] \times [2^{nbR}] \to \mathbb{R}^{a\times b}$ such that for any $A \in \mathbb{R}^{n\times a}$ and $B \in \mathbb{R}^{n\times b}$ with bounded entries we have

$$\mathbb{E}\|A^\top B - g(f_1(A), f_2(B))\|_F^2 < \frac{\|\bar{A}\|_F^2 \|\bar{B}\|_F^2}{n} \left(\frac{2 \cdot 2^{2R} - 1}{(2^{2R} - 1)^2} + \varepsilon\right) + a \cdot b \cdot n^{-8}.$$

Note that the term $\|\bar{A}^\top \bar{B}\|_F^2$ does not appear at all in Theorem 3, and whenever $\|\bar{A}^\top \bar{B}\|^2 \gg \frac{\|\bar{A}\|_F^2 \|\bar{B}\|_F^2}{n}$ the error in Theorem 3 is significantly smaller than the error in Theorem 1.

The scheme used for proving Theorems 1 and 3 is based on using high-dimensional nested lattices with some asymptotically optimal properties. Unfortunately, such lattices do not lend themselves to efficient implementation. In the full version [1] we develop a simplified nested-lattice quantization scheme, based on Conway and Sloane's Voronoi codes [31], that is similar to the one used in the proofs of Theorem 1 and Theorem 3, but uses low-dimensional nested lattices. For such lattices, we suggest a fast implementation, whose computational efficiency does not depend on R. This simplified scheme attains performance fairly close to theoretical estimates therein. In [32] it is illustrated that the fast implementation, which we refer to as NestQuant, attains state-of-the-art result for quantized LLMs. Fast decoding algorithms based on lookup-tables are further explored in [33].

Additional contributions of the full version [1] include the following:

- We study the inner product case $a = b = 1$, in full generality, assuming the entries of A are drawn iid from distribution P, the entries of B are drawn iid from distribution Q, and the rates R_1 and R_2 are not necessarily equal. We derive several upper and lower bounds on the smallest attainable distortion in computing the inner product, and prove some results on the structure of the optimal encoders and decoder.
- For the matrix multiplication case, when the entries of A and B are drawn iid from a distribution P with zero mean and variance σ^2, we show that (4) continues to hold with $\Gamma(R)$ replaced by $\Gamma(R + D(P\|\mathcal{N}(0, \sigma^2))$.

We refer the reader to [1] for full details and proofs, and only sketch the key ideas in Section III.

III. Sketch of the proof

This work started with the goal of trying to understand approximate matrix multiplication for two matrices A and B which are random, with iid Gaussian entries $\mathcal{N}(0, 1)$. We started by trying to solve the case of $a = b = 1$, i.e. when $A^\top B$ is simply an inner product of two iid Gaussian vectors.

Recall that the Gaussian distortion-rate function is $D(R) = 2^{-2R}$, e.g. [34, Section 26.1.2]. A simple argument shows that compressing A to $\hat{A}$ and B to $\hat{B}$ via rate-R optimal Gaussian vector quantizer achieves error

$$\mathbb{E}[(\hat{A}^\top \hat{B} - A^\top B)^2] \le \phi(D(R)), \qquad \phi(x) := 2x - x^2\,.$$

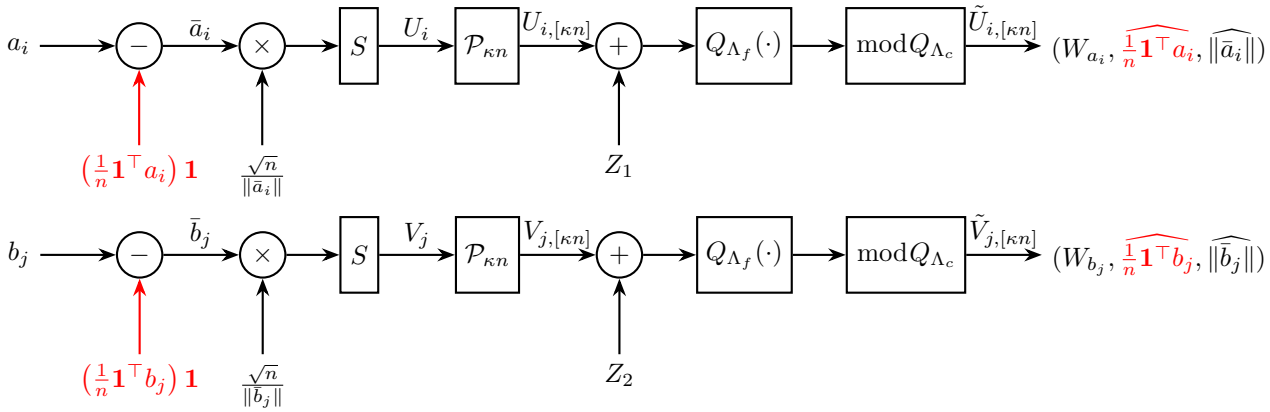

Fig. 1: Encoders for matrix multiplication. Each column of A is encoded by the same encoder, and each column of B is encoded by the same encoder. The encoder used for columns of A and that used for columns of B are also the same, except that for A we use the dither vector $Z_1 \in \mathbb{R}^{\kappa n}$, whereas for B we use the dither vector $Z_2 \in \mathbb{R}^{\kappa n}$. We illustrate the operation of the encoders on the ith column of A, $a_i \in \mathbb{R}^n$, and on the jth column of B, $b_j \in \mathbb{R}^n$. The block S corresponds to left multiplication by the rotation matrix $S \in \mathbb{R}^{n\times n}$, and the block $\mathcal{P}_{\kappa n}$ corresponds to projecting the vector $U_i \in \mathbb{R}^n$ (respectively $V_j \in \mathbb{R}^n$) to $\mathbb{R}^{\kappa n}$, $\kappa \in \frac{1}{n}\cdot\{0,1,\ldots,n\}$, by keeping only its first κn coordinates. Here, κ is the time-sharing/sparsification parameter, determining the fraction of coordinates in each vector that are actually "described" to the decoder. The lattices $\Lambda_c \subset \Lambda_f \subset \mathbb{R}^{\kappa n}$ are nested. The component $Q_{\Lambda_f}(\cdot)$ is a lattice quantizer which maps a point in $\mathbb{R}^{\kappa n}$ to the closest lattice point in Λ_f. The component $\mathrm{mod}\Lambda_c$ maps a point $x \in \mathbb{R}^{\kappa n}$ to $x - Q_{\Lambda_c}(x) \in \mathcal{V}_c$, where $\mathcal{V}_c$ is the Voronoi region of Λ_c. The binary representation W_{a_i} (respectively W_{b_j}) is an encoding of $\tilde{U}_{i,[\kappa n]} \in (\Lambda_f \cap \mathcal{V}_c) \cong \Lambda_f/\Lambda_c$ (respectively $\tilde{V}_{j,[\kappa n]} \in \Lambda_f/\Lambda_c$) using $\log|\Lambda_f/\Lambda_c|$ bits. The scalars $\widehat{\frac{1}{n}\mathbf{1}^\top a_i}, \widehat{\|\bar{a}_i\|}$ (respectively, $\widehat{\frac{1}{n}\mathbf{1}^\top b_j}, \widehat{\|\bar{b}_j\|}$) are high-resolution descriptions of $\frac{1}{n}\mathbf{1}^\top a_i, \|\bar{a}_i\|$ (respectively, $\frac{1}{n}\mathbf{1}^\top b_j, \|\bar{b}_j\|$), which require only $O(\log n)$ bits. The dither vectors Z_1, Z_2 must be known to the decoder. They can be randomly drawn by the encoders and decoder and require sharing randomness between them (in practice, we just store random seed with the matrices). The matrix S need not be known by the decoder. The operations marked in red corresponds to zero-centering the column vectors, and may be avoided altogether. The effect of avoiding those operations on the performance is replacing $\bar{A}$ with A and $\bar{B}$ with B in the MSE upper bounds in Theorems 1 and 3.

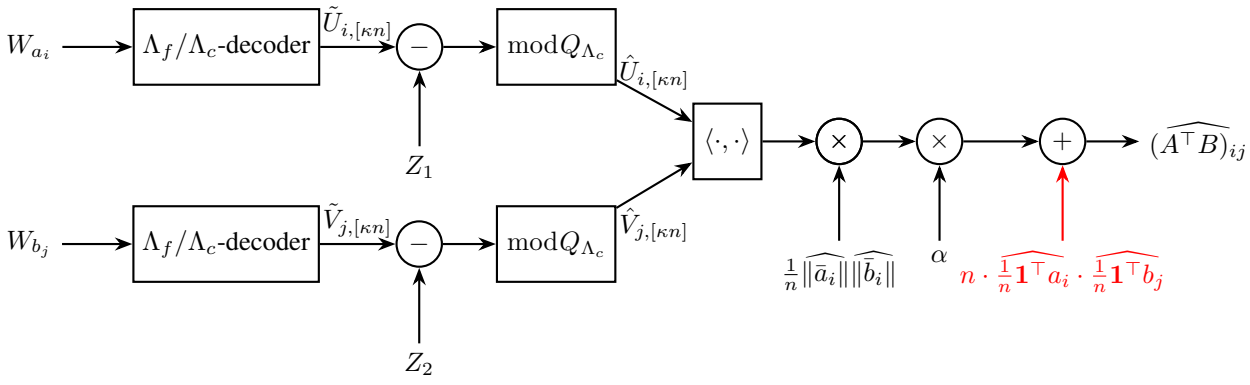

Fig. 2: Decoder for the matrix multiplication problem. We illustrate the estimation of $(A^\top B)_{ij}$. The component Λ_f/Λ_c-decoder maps $\log|\Lambda_f/\Lambda_c|$ bits to points in $\Lambda_f \cap \mathcal{V}_c \subset \mathbb{R}^{\kappa n}$, where $\mathcal{V}_c$ is the Voronoi region of the lattice Λ_c. The component $\langle\cdot,\cdot\rangle$ computes the inner product $\hat{U}_{i,[\kappa n]}^\top \hat{V}_{j,[\kappa n]}$, and $\alpha \in [0,1]$ is a (MMSE-like) scaling coefficient. The operation marked in red need only be implemented if the encoders implemented the corresponding zero-centering operations marked in red in Figure 1. Note that we can estimate the entire product $A^\top B$ by first decoding $\hat{\tilde{A}} = [\hat{U}_{1,[\kappa n]}|\cdots|\hat{U}_{a,[\kappa n]}]$ and $\hat{\tilde{B}} = [\hat{V}_{1,[\kappa n]}|\cdots|\hat{V}_{b,[\kappa n]}]$, computing the matrix $\alpha\hat{\tilde{A}}^\top\hat{\tilde{B}}$, and then computing its Kronecker product with the rank-1 matrix N whose ijth entry is $N_{ij} = \frac{1}{n}\widehat{\|\bar{a}_i\|}\widehat{\|\bar{b}_j\|}$, and adding to it the rank 1 matrix μ whose ijth entry is $\mu_{ij} = n \cdot \widehat{\frac{1}{n}\mathbf{1}^\top a_i} \cdot \widehat{\frac{1}{n}\mathbf{1}^\top b_j}$.

It turned out that the function $\phi(D(R))$ is monotonically decreasing but *not* convex. Thus, via time-sharing one can achieve a lower convex envelope of $\phi(D(R))$, which turns out to be the $\Gamma(R)$ function defined in (2).

We next proceed to lower bounds on distortion or, equivalently, to lower bounds on rate R required for the existence of encoders f_1, f_2 and decoder g satisfying

$$\mathbb{E}[(g(f_1(A), f_2(B)) - A^\top B)^2] \le nD \tag{5}$$

A simple oracle bound (by revealing B to the decoder) shows that rate R cannot be smaller than the standard Shannon rate-distortion function of A. However, this bound leaves a wide gap with the achievability bound given above. Next, by a standard data-processing argument (and observation that encoders for A and B can be without loss of generality be taken identical) we deduce that (5) requires rate

$$R \ge \limsup_{n\to\infty} \frac{1}{n} \inf_{\hat{A}} \{I(A;\hat{A}) : \frac{1}{n}\sum_{i=1}^{n} \phi(\lambda_i) \le D\}\,, \tag{6}$$

where $A \sim \mathcal{N}(0, I_n)$, infimum is over all $\mathbb{R}^n$-valued random variables $\hat{A}$ and $\{\lambda_i\}$ are the eigenvalues of $\mathrm{Cov}(A|\hat{A})$. This reduces inner product quantization to an optimization of a multi-letter mutual information. Notice that the distortion constraint is no longer separable, and hence the standard single-letterization (e.g. [34, Theorem 24.8]) does not work and the limit on the right-hand side is not possible to evaluate. For the special case of Gaussian distribution of entries of A we were able to single-letterize the expression on the right-hand side of (6), showing that left-hand side of (6) evaluates to $\Gamma^{-1}(D)$. Putting both upper and lower bounds together, we conclude that optimal compression rate for the iid Gaussian inner product problem is thus given by $\Gamma^{-1}(D)$.

We next proceed to solving the matrix case. Luckily, it turns out that for Gaussian iid matrices, again, the optimal compression for matrix multiplication of $A^\top B$ is asymptotically achieved by compressing each column separately via the use of optimal inner product quantizers.

Having solved the iid Gaussian case, we proceed to analyzing general (non-random) matrices and vectors. Specifically, for the inner product problem we first normalize each of the two vectors to have norm $\sqrt{n}$ and these norms are compressed using a separate high-resolution scalar quantizer. Next, normalized vectors are multiplied by a common random orthogonal matrix. This makes each resulting vector uniformly distributed on the sphere of radius $\sqrt{n}$, while their inner product is unchanged. As is well known, a high-dimensional vector that is uniform on the sphere is very similar to an iid Gaussian vector (for example, in terms of joint distribution of small $O(\sqrt{n})$-sized subsets). Thus, we reduce the problem to (5) except this time $A_i, B_i \overset{iid}{\sim} \mathcal{N}\left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right)$, where $\rho = \frac{A^\top B}{\|A\|\cdot\|B\|}$. This slight change creates a crucial complication compared to the previous case of $\rho = 0$.

Indeed, suppose we are only tasked with quantizing B and A is given to the decoder undistorted. Because of dependence between two terms in the product $A^\top(B - \hat{B})$ we have to recourse to something like Cauchy-Schwarz, yielding

$$\mathbb{E}[(A^\top B - A^\top \hat{B})^2 \le \mathbb{E}[\|A\|^2\|B - \hat{B}\|^2] = \Omega(n^2)\,.$$

Thus, using "black box" quantizers for A and B only yields n^2 performance guarantees violating (5). This is where *lattice quantization* comes in. Specifically, using the idea of dithering we can make a (randomized) quantizer whose quantization error $(B - \hat{B})$ becomes independent of B and A.

In order to guarantee finite quantization rate, we also need to "truncate" the infinite lattice, for which we use another key idea: a "good" nested lattice quantizer as in [35]–[37]. However, due to the nature of the problem we require construction of nested lattice pairs that satisfy stronger conditions than were known from prior work. In particular, building upon the heavy-lifting in a recent [38], we show that for most lattices the spectrum of the quantization error is nearly "white", in a suitable sense. Overall, we construct quantizers for inner product problem of non-random vectors with a reconstruction error that depends only on the inner product between the vectors and their individual ℓ_2 norms. Since the performance bounds coincides with the lower bound for the iid Gaussian case, it turns out that the resulting quantizers are optimal and generally cannot be improved (except, possibly, in terms of finite-n performance). Together these steps complete proof of the main results quoted above.

Remark on ϵ-nets and randomization via rotation (and dithering). We believe that the effect of randomization is crucial to our construction. Indeed, consider the special case of $a = b = 1$ and vectors A, B constrained to be norm $\|A\| = \|B\| = \sqrt{n}$. Suppose for simplicity that vector B is allowed to be quantized at infinite rate and we are only interested in quantizing A to nR bits. With this budget, the standard idea would be to create an $O(\sqrt{n})$-net covering the $\sqrt{n}\mathbb{S}^{n-1}$ and set $\hat{A}$ to be the nearest neighbor in this net. What performance can this scheme guarantee? Since A and B can be arbitrary the best we can do to is a Cauchy-Schwarz estimate

$$(A^\top B - \hat{A}^\top B)^2 \le \|A - \hat{A}\|^2\|B\|^2 \asymp n^2\,.$$

Thus, whereas our lattice quantizer yields guarantee $O(n)$ on quadratic error for the inner product, the trivial ϵ-net argument (even with B given for free) only yields n^2 bound. As we described above, the key benefit of rotation, complemented by dithering, is making $A - \hat{A}$ a zero-mean vector.

ACKNOWLEDGMENT

The work of OO was supported by the Israel Science Foundation (ISF), grant No. 1641/21. The work of YP was supported in part by the MIT-IBM Watson AI Lab and by the National Science Foundation under Grant No CCF-2131115.

REFERENCES

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

[2] V. Strassen, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[3] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[4] R. Duan, H. Wu, and R. Zhou, "Faster matrix multiplication via asymmetric hashing," in *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2023, pp. 2129–2138.

[5] V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, *New Bounds for Matrix Multiplication: from Alpha to Omega*, 2024, pp. 3792–3835.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[7] M. W. Mahoney *et al.*, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[8] P.-G. Martinsson and J. A. Tropp, "Randomized numerical linear algebra: Foundations and algorithms," *Acta Numerica*, vol. 29, pp. 403–572, 2020.

[9] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices i: Approximating matrix multiplication," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 132–157, 2006.

[10] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380–388.

[11] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262.

[12] R. Pagh, "Compressed matrix multiplication," *ACM Transactions on Computation Theory (TOCT)*, vol. 5, no. 3, pp. 1–17, 2013.

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

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2018.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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

[25] D. Malak, "Distributed structured matrix multiplication," *arXiv preprint arXiv:2405.02904*, 2024.

[26] D. Krithivasan and S. S. Pradhan, "Lattices for distributed source coding: Jointly gaussian sources and reconstruction of a linear function," *IEEE Transactions on Information Theory*, vol. 55, no. 12, pp. 5628–5651, 2009.

[27] A. B. Wagner, "On distributed compression of linear functions," *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 79–94, 2010.

[28] A. Ingber, T. Courtade, and T. Weissman, "Compression for quadratic similarity queries," *IEEE transactions on information theory*, vol. 61, no. 5, pp. 2729–2747, 2015.

[29] I. Ochoa, A. Ingber, and T. Weissman, "Compression schemes for similarity queries," in *2014 Data Compression Conference*, 2014, pp. 332–341.

[30] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[31] 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.

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

[33] I. Kaplan and O. Ordentlich, "High-rate nested-lattice quantized matrix multiplication with small lookup tables," in *Proc. ISIT 2025*, Ann Arbor, Michigan, June 2025.

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

[35] U. Erez and R. Zamir, "Achieving 1/2 log (1+ snr) on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2293–2314, 2004.

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

[37] O. Ordentlich and U. Erez, "A simple proof for the existence of "good" pairs of nested lattices," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4439–4453, 2016.

[38] O. Ordentlich, O. Regev, and B. Weiss, "New bounds on the density of lattice coverings," *Journal of the American Mathematical Society*, vol. 35, no. 1, pp. 295–308, 2022.