Accelerating Large-Scale Inference with Anisotropic Vector Quantization

Ruiqi Guo [*1] **Philip Sun** [*1] **Erik Lindgren** [*1] **Quan Geng** [1] **David Simcha** [1] **Felix Chern** [1] **Sanjiv Kumar** [1]

Abstract

Quantization based techniques are the current state-of-the-art for scaling maximum inner product search to massive databases. Traditional approaches to quantization aim to minimize the reconstruction error of the database points. Based on the observation that for a given query, the database points that have the largest inner products are more relevant, we develop a family of anisotropic quantization loss functions. Under natural statistical assumptions, we show that quantization with these loss functions leads to a new variant of vector quantization that more greatly penalizes the parallel component of a datapoint's residual relative to its orthogonal component. The proposed approach, whose implementation is open-source, achieves state-of-the-art results on the public benchmarks available at `ann-benchmarks.com`.

1. Introduction

Maximum inner product search (MIPS) has become a popular paradigm for solving large scale classification and retrieval tasks. For example, in recommendation systems, user queries and documents are embedded into a dense vector space of the same dimensionality and MIPS is used to find the most relevant documents given a user query (Cremonesi et al., 2010). Similarly, in extreme classification tasks (Dean et al., 2013), MIPS is used to predict the class label when a large number of classes, often on the order of millions or even billions are involved. Lately, MIPS has also been applied to training tasks such as scalable gradient computation in large output spaces (Yen et al., 2018), efficient sampling for speeding up softmax computation (Mussmann and Ermon, 2016) and sparse updates in end-to-end trainable memory systems (Pritzel et al., 2017).

To formally define the Maximum Inner Product Search (MIPS) problem, consider a database $X = \{x_i\}_{i=1,2,\ldots,n}$ with n datapoints, where each datapoint $x_i \in \mathbb{R}^d$ in a d-dimensional vector space. In the MIPS setup, given a query $q \in \mathbb{R}^d$, we would like to find the datapoint $x \in X$ that has the highest inner product with q, i.e., we would like to identify

$$x_i^* := \arg\max_{x_i \in X} \langle q, x_i \rangle.$$

Exhaustively computing the exact inner product between q and n datapoints is often expensive and sometimes infeasible. Several techniques have been proposed in the literature based on hashing, graph search, or quantization to solve the approximate maximum inner product search problem efficiently, and the quantization based techniques have shown strong performance (Ge et al., 2014; Babenko and Lempitsky, 2014; Johnson et al., 2017).

In most traditional quantization works, the objective in the quantization procedures is to minimize the reconstruction error for the database points. We show this is a suboptimal loss function for MIPS. This is because for a given query, quantization error for database points that score higher, or have larger inner products, is more important. Using this intuition, we propose a new family of score-aware quantization loss functions and apply it to multiple quantization techniques. Our contributions are as follows:

- We propose the score-aware quantization loss function. The proposed loss can work under any weighting function of the inner product and regardless of whether the datapoints vary in norm.
- Under natural statistical assumptions, we show that the score-aware quantization loss can be efficiently calculated. The loss function leads to an anisotropic weighting that more greatly penalizes error parallel with the datapoint than error orthogonal to the datapoint.
- The proposed loss is generally applicable to many quantization methods. We demonstrate the codebook learning and quantization procedures for product quantization and vector quantization can be efficiently adapted to the proposed loss function.
- We show that anisotropic quantization leads to large MIPS performance gains over reconstruction loss-

[*]Equal contribution [1]Google Research. Correspondence to: Philip Sun <sunphil@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

based techniques. Our method achieves state-of-the-art performance on standard large-scale benchmarks such as Glove-1.2M. In addition to recall gains, anisotropic quantization gives significantly more accurate inner product value approximations.

2. Background and Related Works

2.1. Inference as Maximum Inner Product Search

Efficient maximum inner product search (MIPS) is necessary for many large-scale machine learning systems. One popular approach to information retrieval systems and recommender systems uses *representation learning* in the embedding space. In this framework, we learn embedding functions to map items to be retrieved in a common vector space, where the items can be words, images, users, audio, products, web pages, graph nodes, or anything of interest (Cremonesi et al., 2010; Weston et al., 2010; Guo et al., 2016a; Gillick et al., 2019; Wu et al., 2017).

In recommender systems, two networks are jointly trained to generate query (user) vectors and item vectors, such that embedding vectors of queries and relevant items have high inner product when computed in the embedding space. To perform inference, we first pre-compute a database of embedding vectors for items to be recommended. When a query arrives, we compute the query embedding then return the items with the highest inner product. In extreme classification, a neural network classifier is trained, where each row of the weight matrix of the classification layer corresponds to the embedding of a class label (Dean et al., 2013; Reddi et al., 2019). In both settings, the computationally expensive operation is finding the item embedding that has the largest inner product with the query embedding, which can be efficiently solved by Maximum Inner Product Search (MIPS).

2.2. Methods for accelerating MIPS

There is a large body of similarity search literature on max inner product and nearest neighbor search. We refer readers to (Wang et al., 2014; 2016) for a comprehensive survey. We include a brief summary here.

There are two main tasks required to develop an efficient MIPS system. One task is to reduce the number of items that are scored to identify the top result. This is typically done with a space partitioning method. The other task is improving the rate at which items are scored. This is typically done with quantization, and is where the main contribution of our work lies. Successful implementation of MIPS systems requires good performance in both tasks.

Many researchers have developed high quality implementations of libraries for nearest neighbor search, such as SPTAG (Chen et al., 2018), FAISS (Johnson et al., 2017), and hnswlib (Malkov and Yashunin, 2016). We compare with the ones available on ANN-Benchmarks in Section 5.

2.2.1. Reducing the Number of Evaluations

One class of approaches to reducing the number of items scored is space partitioning. These approaches partition the space into different buckets. To perform MIPS in this setting, we find the relevant buckets for a given query and score only the items in these buckets.

Examples of this approach include tree search methods and locality sensitive hashing. Tree search methods such as (Muja and Lowe, 2014; Dasgupta and Freund, 2008) partition the space recursively, forming a tree. Locality sensitive hashing (Shrivastava and Li, 2014; Neyshabur and Srebro, 2015; Indyk and Motwani, 1998; Andoni et al., 2015) partitions the space using a similarity-preserving hash function. There is also a class of approaches based on graph search (Malkov and Yashunin, 2016; Harwood and Drummond, 2016). These methods work by navigating a graph by greedily selecting the neighbor with the highest dot product.

2.2.2. Quantization

Quantization is an important technique for building state-of-the-art MIPS systems in large scale settings. Below we describe the several ways that quantization improves performance.

- Efficient dot product computations: We can calculate the dot product of a d dimensional query vector with n quantized points in time $O(dk + mn)$ using look up tables, where k is the size of each quantization codebook and m is the number of codebooks. For typical choices of k and m this is faster than the $O(nd)$ complexity required for exact computation.
- Memory bandwidth: modern processors need workloads with a high amount of computation per memory read in order to fully utilize their resources. Quantization compresses datapoints, resulting in less memory bandwidth usage and higher processor utilization.
- Storage: quantized datapoints take up less space in memory or on disk. For large-scale datasets, this allows more datapoints to be stored on a single machine.

One approach to quantization is with random projections (Charikar, 2002; Vempala, 2005; Li and Li, 2019). One issue with random projections is that quantization is oblivious to the data, and it may be more efficient to use a quantization method that is able to exploit structure in the data. Quantization methods of this form are available for binary quantization (He et al., 2013; Liong et al., 2015; Dai et al., 2017),

product quantization (Jegou et al., 2011; Guo et al., 2016b; Zhang et al., 2014; Wu et al., 2017), additive quantization (Babenko and Lempitsky, 2014; Martinez et al., 2018), and ternary quantization (Zhu et al., 2016). We discuss product quantization in more detail in Section 4. There are also lines of work that focus on learning transformations before quantization (Gong et al., 2013; Ge et al., 2014; Sablayrolles et al., 2019). Learning quantization from the observed data distribution also has been studied in (Marcheret et al., 2009; Morozov and Babenko, 2019; Babenko et al., 2016).

Our work differs from the above methods as they essentially focus on minimizing reconstruction error as a loss function, while we develop an approach in the following section where we minimize a novel loss function that is designed to improve the downstream MIPS objective.

We also highlight the work May et al. (2019), where they consider quantization objectives for word embeddings that improve the downstream performance of training models for natural language processing tasks.

3. Problem Formulation

Common quantization techniques focus on minimizing the reconstruction error (sum of squared error) when x is quantized to $\tilde{x}$. It can be shown that minimizing the reconstruction errors is equivalent to minimizing the expected inner product quantization error under a mild condition on the query distribution without assumption on the database point distribution. Indeed, consider the quantization objective of minimizing the expected total inner product quantization errors over the query distribution:

$$\mathbb{E}_q \sum_{i=1}^{n} (\langle q, x_i \rangle - \langle q, \tilde{x_i} \rangle)^2 = \mathbb{E}_q \sum_{i=1}^{n} \langle q, x_i - \tilde{x_i} \rangle^2. \quad (1)$$

Under the assumption that q is isotropic, i.e., $\mathbb{E}[qq^T] = cI$, where I is the identity matrix and $c \in \mathbb{R}^+$, the objective function becomes

$$\begin{aligned} \sum_{i=1}^{n} \mathbb{E}_q \langle q, x_i - \tilde{x_i} \rangle^2 &= \sum_{i=1}^{n} \mathbb{E}_q (x_i - \tilde{x_i})^T qq^T (x_i - \tilde{x_i}) \\ &= c \sum_{i=1}^{n} \|x_i - \tilde{x_i}\|^2 \end{aligned}$$

Therefore, the objective becomes minimizing the reconstruction errors of the database points $\sum_{i=1}^{n} \|x_i - \tilde{x_i}\|^2$, and this has been considered extensively in the literature.

One key observation about the above objective function (1) is that it takes expectation over all possible combinations of datapoints x and queries q. However, it is easy to see that not all pairs of (x, q) are equally important. The approximation error on the pairs which have a high inner product is far more important since they are likely to be among the top ranked pairs and can greatly affect the search result, while for the pairs whose inner product is low the approximation error matters much less. In other words, for a given datapoint x, we should quantize it with a bigger focus on its error with those queries which have high inner product with x. See Figure 1 for the illustration.

Following this key observation, we propose the *score-aware quantization loss*. This is a new loss function for quantization that weighs the inner product approximation error by w, an arbitrary function of our choice that returns a weight based on the value of the true inner product. Specifically, we define the loss function as the following:

Definition 3.1. Given a datapoint x_i, its quantization $\tilde{x_i}$, and a weight function $w : \mathbb{R} \mapsto \mathbb{R}^+$ of the inner product score, the *score-aware quantization* loss with respect to a query distribution $\mathcal{Q}$ is defined as

$$\ell(x_i, \tilde{x_i}, w) = \mathbb{E}_{q \sim \mathcal{Q}}[w(\langle q, x_i \rangle) \langle q, x_i - \tilde{x_i} \rangle^2]. \quad (2)$$

Since the norm of q does not matter to the ranking result, we can assume $||q|| = 1$ without loss of generality. Similarly, assuming we have no prior knowledge of the query distribution $\mathcal{Q}$, we trivially assume q is uniformly spherically distributed. The expectation can be recomputed if $\mathcal{Q}$ is known or estimated empirically.

3.1. Analyzing Score-Aware Quantization Loss

We show that regardless of the choice of w, a score-aware quantization loss $\ell(x_i, \tilde{x_i}, w)$ always decomposes into an anisotropic weighted sum of the magnitudes of the parallel and orthogonal residual errors. These two errors are defined as follows: first, define the residual error of a quantization $\tilde{x_i}$ as $x_i - \tilde{x_i}$. The *parallel residual error* is the component of the residual error parallel to the datapoint x_i; it can be computed as

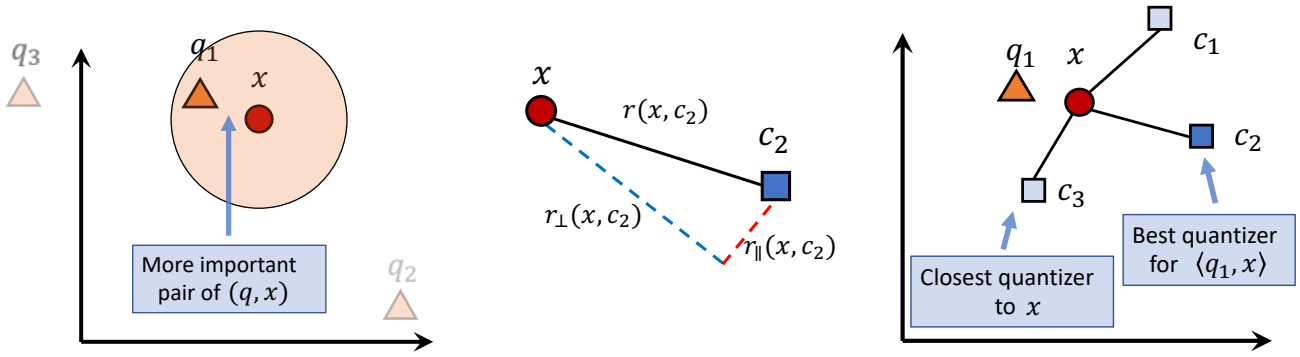
$$r_{\parallel}(x_i, \tilde{x_i}) = \frac{\langle (x_i - \tilde{x_i}), x_i \rangle x_i}{||x_i||^2}.$$

Orthogonal residual error is defined analogously, and can be computed as

$$r_{\perp}(x_i, \tilde{x_i}) = (x - x_i) - r_{\parallel}(x_i, \tilde{x_i}).$$

These two components are illustrated in Figure 1b. The relative weights of these two error components in contributing to the score-aware loss are determined by the choice of w.

Theorem 3.2. *Suppose we are given a datapoint x_i, its quantization $\tilde{x_i}$, and a weight function w. Assuming that query q is uniformly distributed in the d-dimensional unit sphere, the score-aware quantization loss equals*



(a) (b) (c)

Figure 1. (a) Not all pairs of q and x are equally important: for x, it is more important to accurately quantize the inner product of $\langle q_1, x \rangle$ than $\langle q_2, x \rangle$ or $\langle q_3, x \rangle$, because $\langle q_1, x \rangle$ has a higher inner product and thus is more likely to be the maximum; (b) Quantization error of x given one of its quantizer c_2 can be decomposed to a parallel component $r_\parallel$ and an orthogonal component $r_\perp$. (c) Graphical illustration of the intuition behind Equation (2). Even if c_3 is closer to x in terms of Euclidean distance, c_2 is a better quantizer than c_3 in terms of inner product approximation error of $\langle q_1, x - c \rangle$. Notice that c_3 incur more parallel loss ($r_\parallel$), while c_2 incur more orthogonal loss ($r_\perp$).

$$\ell(x_i, \tilde{x_i}, w) = h_\parallel(w, ||x_i||)||r_\parallel(x_i, \tilde{x_i})||^2 + h_\perp(w, ||x_i||)||r_\perp(x_i, \tilde{x_i})||^2$$

with $h_\parallel$ and $h_\perp$ defined as follows:

$$h_\parallel := (d-1)\int_0^\pi w(||x_i||\cos\theta)(\sin^{d-2}\theta - \sin^d\theta)d\theta$$
$$h_\perp := \int_0^\pi w(||x_i||\cos\theta)\sin^d\theta d\theta.$$

Proof. See Appendix Section 7.1. □

Any weight function would work for the above proposed loss. For the MIPS problem, it is intuitive to choose w so that it puts greater weight on larger inner products. For such w, we show that parallel quantization error is weighted more heavily than orthogonal quantization error. This is formalized below and illustrated in Figure 1.

Theorem 3.3. *For any w such that $w(t) = 0$ for $t < 0$ and $w(t)$ is monotonically non-decreasing for $t \geq 0$,*

$$h_\parallel(w, ||x_i||) \geq h_\perp(w, ||x_i||)$$

with equality if and only if $w(t)$ is constant for $t \in [-||x_i||, ||x_i||]$.

Proof. See Appendix Section 7.2. □

3.2. Special case of $w(t) = \mathbf{I}(t \geq T)$

One particular w of interest is the function $w(t) = \mathbf{I}(t \geq T)$. This weight function only considers quantization loss when the dot product is above a threshold T. Since $\mathbf{I}(t \geq T)$ satisfies the conditions for Theorem 3.3, it effectively penalizes parallel quantization error more greatly than orthogonal error. With this weight function, our expressions for $h_\parallel$ and $h_\perp$ simplify to:

$$h_\parallel = (d-1)\int_0^{\arccos(T/||x_i||)} \sin^{d-2}\theta - \sin^d\theta d\theta$$
$$h_\perp = \int_0^{\arccos(T/||x_i||)} \sin^d\theta d\theta$$

With $w(t) = \mathbf{I}(t \geq T)$, we have

$$\begin{aligned}\ell(x_i, \tilde{x}_i, w) &= h_\parallel(w, ||x_i||)||r_\parallel(x_i, \tilde{x}_i)||^2 + \\ &\quad h_\perp(w, ||x_i||)||r_\perp(x_i, \tilde{x}_i)||^2 \\ &\propto \eta(w, ||x_i||)||r_\parallel(x_i, \tilde{x}_i)||^2 + ||r_\perp(x_i, \tilde{x}_i)||^2\end{aligned}$$

where $\eta(w, ||x_i||) := \dfrac{h_\parallel(w, ||x_i||)}{h_\perp(w, ||x_i||)}$.

We can recursively compute $\eta(w = \mathbf{I}(t \geq T), ||x_i||)$ as a function of d analytically. Furthermore we can prove that $\frac{\eta}{d-1}$ has an limit as $d \to \infty$, as demonstrated empirically in Figure 2. We can use this limit, which is easy to evaluate, as a proxy of η in computing the proposed loss.

Theorem 3.4.

$$\lim_{d\to\infty} \frac{\eta(\mathbf{I}(t \geq T), ||x_i||)}{d-1} = \frac{(T/||x_i||)^2}{1 - (T/||x_i||)^2} \quad (3)$$

Proof. See Appendix Section 7.3. □

As special cases, when $T = 0$, $\eta(\mathbf{I}(t \geq 0), ||x_i||) = 1$ which implies parallel and orthogonal errors are weighted

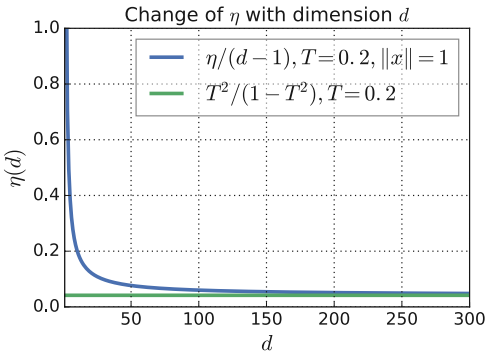


Figure 2. The ratio $\eta(\mathbf{I}(t \geq T = 0.2), \|x\| = 1)/(d-1)$ in Theorem 3.4 computed analytically as a function of d quickly approaches the limit defined in Equation (3).

equally. When $T = |||x_i||$, we have $\eta(\mathbf{I}(t \geq ||x_i||), ||x_i||) = \infty$ which indicates we should only consider parallel error.

Theorem 3.2 shows that the weight of each datapoint's parallel and orthogonal quantization errors are dependent on $||x_i||$. However, when the database has constant norm, i.e. $||x_i|| = c$, we can use the following simplified form:

$$\sum_{i=1}^{n} \ell(x_i, \tilde{x}_i, \mathbf{I}(t \geq T))$$
$$\propto \eta(w, c) \sum_{i=1}^{n} ||r_{\parallel}(x_i, \tilde{x}_i)||^2 + \sum_{i=1}^{n} ||r_{\perp}(x_i, \tilde{x}_i)||^2$$

4. Application to Quantization Techniques

In this section we consider the codebook learning and quantization procedure for our proposed anisotropic loss function. In the previous sections, we established that the loss function, $\ell(x_i, \tilde{x}_i, w)$ leads to a weighted combination of parallel quantization error and orthogonal quantization error. In practice, we can choose a fixed η according to the choice of w such as the one suggested in Section 3.2.

In vector quantization, we first construct a dictionary $C = \{c_1, c_2, \ldots, c_k\}$. To quantize a vector x we replace x with one of the codewords. Typically, the quantized vector $\tilde{x}$ minimizes some loss function: $\tilde{x} = \arg\min_{c_1, c_2, \ldots, c_k} L(x_i, c_i)$.

After we quantize a database of n points, we can calculate the dot product of a query vector q with all quantized points in $O(kd + n)$ time. This is much better than the $O(nd)$ time required for the original unquantized database. We achieve the $O(kd+n)$ runtime by computing a lookup table containing the inner product of the q with each of the k codewords in $O(kd)$ time. We then do a table lookup for each of the n datapoints to get their corresponding inner products.

In order to construct the dictionary C, we need to optimize the choice of codewords over the loss function. For ℓ_2-reconstruction loss, the optimization problem becomes

$$\min_{c_1, c_2, \ldots, c_k \in \mathbb{R}^d} \sum_{x_i} \min_{\tilde{x}_i \in \{c_1, c_2, \ldots, c_k\}} \|x_i - \tilde{x}_i\|^2.$$

This is exactly the well-studied k-means clustering objective, which is often solved using Lloyd's algorithm.

If, as in the previous section, we have our loss function $\ell(x, \tilde{x}) = h_{i,\parallel}\|r_{\parallel}(x_i, \tilde{x_i})\|^2 + h_{i,\perp}\|r_{\perp}(x_i, \tilde{x_i})\|^2$ for appropriate scaling parameters $h_{i,\parallel}$, $h_{i,\perp}$, we obtain a new objective function we call the *anisotropic vector quantization problem*.

Definition 4.1. Given a dataset $x_1, x_2, \ldots, x_n$ of points in $\mathbb{R}^d$, scaling parameters $h_{i,\parallel}$, $h_{i,\perp}$ for every datapoint x_i, and k codewords, the *anisotropic vector quantization problem* is finding the k codewords that minimize the objective function

$$\min_{c_1, \ldots, c_k} \sum_{x_i} \min_{\tilde{x}_i \in \{c_1, \ldots, c_k\}} h_{i,\parallel}\|r_{\parallel}(x_i, \tilde{x}_i)\|^2 + h_{i,\perp}\|r_{\perp}(x_i, \tilde{x}_i)\|^2.$$

Next we develop an iterative algorithm to optimize the anisotropic vector quantization problem. Similar to Lloyd's algorithm (Lloyd, 1982), our algorithm iterate between partition assignment step and codebook update step:

1. (Initialization Step) Initialize codewords $c_1, c_2, \ldots, c_k$ to be random datapoints sampled from $x_1 \ldots x_n$.
2. (Partition Assignment Step) For each datapoint x_i find its codeword $\tilde{x_i} = \arg\min_{\tilde{x}_i \in \{c_1, \ldots, c_k\}} \ell(x_i, \tilde{x}_i)$. This can be done by enumerating all k possile choices of codewords.
3. (Codebook Update Step) For every codeword c_j, let X_j be all datapoints x_i such that $\tilde{x}_i = c_j$. Update c_j by
$$c_j \leftarrow \arg\min_{c \in \mathbb{R}^d} \sum_{x_i \in X_j} \ell(x_i, c).$$
4. Repeat Step 2 and Step 3 until convergence to a fixed point or maximum number of iteration is reached.

In each iteration, we need perform update step for each of the codeword. Given a partition of the datapoints X_j, we can find the optimal value of the codeword c_j that minimizes the following objective:

$$c_j = \arg\min_{c \in \mathbb{R}^d} \sum_{x \in X_j} h_{i,\parallel}\|r_{\parallel}(x_i, c)\|^2 + h_{i,\perp}\|r_{\perp}(x_i, c)\|^2. \quad (4)$$

By setting gradient respect to c_j to zero, we obtain the following update rule:

Theorem 4.2. *Optimal codeword c_j can be obtained in closed form by solving the optimization problem in Equation (4) for a partition X_j. The update rule for the codebook is*

$$c_j^* = \left(I \sum_{x_i \in X_j} h_{i,\perp} + \sum_{x_i \in X_j} \frac{h_{i,\parallel} - h_{i,\perp}}{||x_i||^2} x_i x_i^T \right)^{-1} \sum_{x_i \in X_j} h_{i,\perp} x_i$$

Proof. See Section 7.4 of the Appendix for the proof. □

As expected, we see that when all $h_{i,\parallel} = h_{i,\perp}$, our codeword update is equivalent to finding the weighted average of the partition. Furthermore, if $w(t) = 1$, the update rule becomes finding the average of datapoints in the partition, same as standard k-means update rule. Additionally, since there are only a finite number of partitions and at every iteration the loss function decreases or stays constant, our solution will eventually converge to a fixed point.

4.1. Product Quantization

In vector quantization with a dictionary of size k, we quantize each datapoint into one of k possible codewords. We can think of this as encoding each datapoint with one dimension with k possible states.

With *product quantization* we encode each datapoint into an M dimensional codeword, each with k possible states. This allows us to represent k^M possible codewords, which would not be scalable with vector quantization. To do this, we split each datapoint x into M subspaces each of dimension d/M: $x = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$. We then create M dictionaries $C^{(1)}, C^{(2)}, \ldots, C^{(m)}$, each with k codewords of dimension d/M. Each datapoint would then be encoded with M dimensions, with every dimension taking one of k states.

To calculate distances with product quantization, for every dictionary $C^{(m)}$ we calculate the partial dot product of the relevant subspace of the query with every codeword in the dictionary. The final dot product is obtain by sum up all M partial dot product. We can then calculate the dot product with m quantized datapoints in time $O(kd + mn)$.

Using our anisotropic loss function $\ell(x_i, \tilde{x}_i) = h_{i,\parallel}\|r_\parallel(x_i, \tilde{x}_i)\|^2 + h_{i,\perp}\|r_\perp(x_i, \tilde{x}_i)\|^2$ we obtain a new objective function for product quantization we call the anisotropic product quantization problem.

Definition 4.3. Given a dataset $x_1, x_2, \ldots, x_n$ of points in $\mathbb{R}^d$, a scaling parameter η, a number M of dictionaries each with elements of size d/M and k codewords in each dictionary, the *anisotropic product quantization problem* is to find the M dictionaries that minimizes

$$\min_{\substack{C^{(m)} \subseteq \mathbb{R}^{d/M} \\ |C^{(m)}| = k}} \sum_{x_i} \min_{\tilde{x}_i \in \prod_m C^{(m)}} h_{i,\parallel}\|r_\parallel(x_i, \tilde{x}_i)\|^2 + h_{i,\perp}\|r_\perp(x_i, \tilde{x}_i)\|^2.$$

We again consider an iterative algorithm for the problem. We first initialize all quantized datapoints with some element from every dictionary. We then consider the following iterative procedure:

1. (Initialization Step) Select a dictionary $C^{(m)}$ by sampling from $\{x_1^{(m)}, \ldots x_n^{(m)}\}$.
2. (Partition Assignment Step) For each datapoint x_i, update $\tilde{x}_i$ by using the value of $c \in C^{(m)}$ that minimizes the anisotropic loss of $\tilde{x}_i$.
3. (Codebook Update Step) Optimize the loss function over all codewords in all dictionaries while keeping every dictionaries partitions constant.
4. Repeat Step 2 and Step 3 until convergence to a fixed point or maximum number of iteration is reached.

We can perform the update step efficiently since once the partitions are fixed the update step minimizes a convex loss, similar to that of vector quantization. We include details in Section 7.5 of the Appendix. Additionally, since there are a finite number of partition assignment and at every step the loss function decreases or stays constant, our solution will eventually converge to a fixed point. We note that we can also optionally initialize the codebook by first training the codebook under regular ℓ_2-reconstruction loss, which speed up training process.

5. Experiments

In this section, we show our proposed quantization objective leads to improved performance on maximum inner product search. First, we fix the quantization mechanism and compare traditional reconstruction loss with our proposed loss to show that score-aware loss leads to better retrieval performance and more accurate estimation of maximum inner product values. Next, we compare in fixed-bit-rate settings against QUIPS and LSQ, which are the current state-of-the-art for many MIPS tasks. Finally, we analyze the end-to-end MIPS retrieval performance of our algorithm in terms of its speed-recall trade-off in a standardized hardware environment. We used the benchmark setup from `ann-benchmarks.com`, which provides 11 competitive baselines with pre-tuned parameters. We plot each algorithm's speed-recall curve and show ours achieves the state-of-the-art.

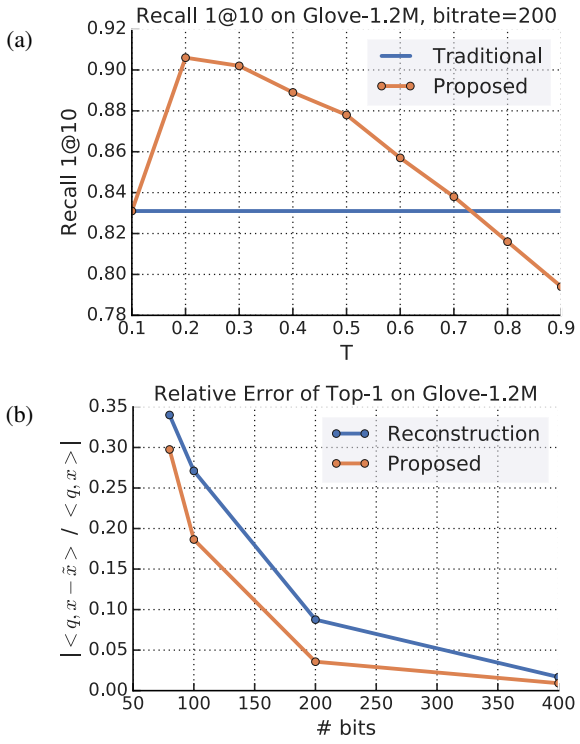


Figure 3. (a) The retrieval Recall1@10 for different values of the threshold T. We see that for $T = 0.2$ (corresponding to $\eta = 4.125$) our proposed score-aware quantization loss achieves significantly better Recall than traditional reconstruction loss. (b) The relative error of inner product estimation for true Top-1 on `Glove1.2M` dataset across multiple number of bits settings. We see that our proposed score-aware quantization loss reduces the relative error compared to reconstruction loss.

5.1. Direct comparison with reconstruction loss

We compare our proposed score-aware quantization loss with the traditional reconstruction loss by fixing all parameters other than the loss function in the following experiments.

We use `Glove1.2M` which is a collection of 1.2 million 100-dimensional word embeddings trained as described in (Pennington et al., 2014). See Section 7.8 of the Appendix for our rationale for choosing this dataset. For all experiments we choose $w(t) = \mathbf{I}(t \geq T)$. The Glove dataset is meant to be used with a cosine distance similarity metric, while our algorithm is designed for the more general MIPS task. MIPS is equivalent to cosine similarity search when all datapoints are equal-norm, so we adopt our technique to cosine similarity search by unit-normalizing all datapoints at training time.

We first compare the two losses by their Recall1@10 when used for product quantization on `Glove1.2M`, as shown in Figure. 3a. We learn a dictionary by optimizing product quantization with reconstruction loss. We then quantize datapoints two ways, first by minimizing reconstruction loss and then by minimizing score-aware loss. We see that score-aware quantization loss achieves significant recall gains as long as T is chosen reasonably. For all subsequent experiments, we set $T = 0.2$, which by the limit in Equation (3) corresponds to a value of $\eta = 4.125$.

Next we look at the accuracy of the estimated top-1 inner product as measured by relative error: $|\frac{\langle q,x\rangle - \langle q,\tilde{x}\rangle}{\langle q,x\rangle}|$. This is important in application scenarios where an accurate estimate of $\langle q, x\rangle$ is needed, such as softmax approximation, where the inner product values are often logits later used to compute probabilities. One direct consequence of score-aware loss functions is that the objective weighs pairs by their importance and thus leads to lower estimation error on top-ranking pairs. We see in Figure. 3b that our score-aware loss leads to smaller relative error over all bitrate settings.

Datasets other than Glove demonstrate similar performance gains from score-aware quantization loss. See Section 7.6 of the Appendix for results on the `Amazon-670k` extreme classification dataset.

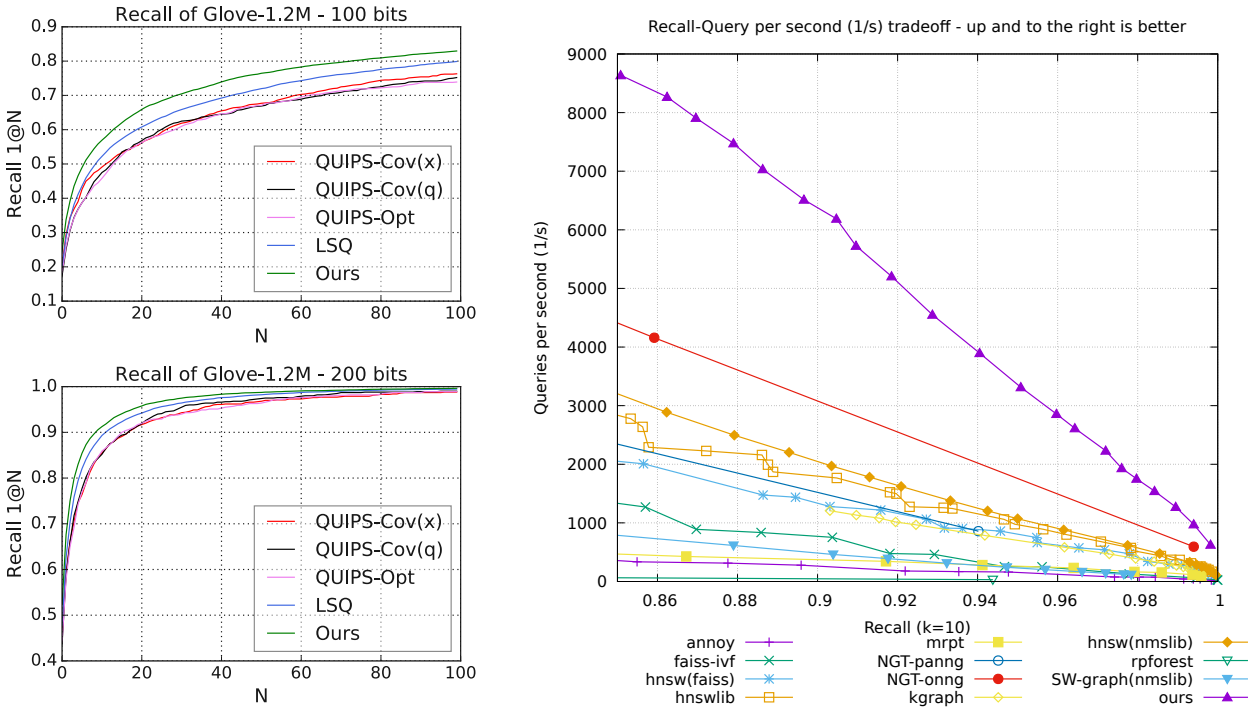
5.2. Maximum inner product search retrieval

Next, we compare our MIPS retrieval performance against other quantization techniques at equal bitrate. We compare to LSQ (Martinez et al., 2018) and all three variants of QUIPS (Guo et al., 2016b). In Figure 4a we measure the performance at fixed bitrates of 100 and 200 bits per datapoint. Our metric is Recall 1@N, which corresponds to the proportion of queries where the top N retrieved results contain the true top-1 datapoint. Our algorithm using score-aware loss outperforms other algorithms at both bitrates and all ranges of N.

Other quantization methods may also benefit from using score-aware quantization loss. For example, binary quantization techniques such as (Dai et al., 2017) use reconstruction loss in their original paper, but can be easily adapted to the proposed loss by a one line change to the loss objective. We show results which illustrate the improvement of such a change in Section 7.7 of Appendix.

5.3. Recall-Speed benchmark

Fixed-bit-rate experiments mostly compare asymptotic behavior and often overlook preprocessing overhead such as learned rotation or lookup table computation, which can be substantial. To evaluate effectiveness of MIPS algorithms in a realistic setting, it is important to perform end-to-end benchmarks and compare speed-recall curves. We adopted the methodology of public benchmark ANN-Benchmarks (Aumüller et al., 2019), which plots a comprehensive set of 11 algorithms for comparison, including



(a) MIPS recall on `Glove1.2M`.

(b) Speed-recall trade-off on `Glove1.2M` Recall 10@10.

Figure 4. (a) Recall 1@N curve on `Glove1.2M` comparing with variants of QUIPS (Guo et al., 2016b) and LSQ (Martinez et al., 2018) on MIPS tasks. We see that our method improves over all of these methods. (b) Recall-Speed benchmark with 11 baselines from (Aumüller et al., 2019) on `Glove1.2M`. The parameters of each baseline are pre-tuned and released on: `http://ann-benchmarks.com/`. We see that our approach is the fastest in the high recall regime.

`faiss` (Johnson et al., 2017) and `hnswlib` (Malkov and Yashunin, 2016).

Our benchmarks are all conducted on an Intel Xeon W-2135 with a single CPU thread, and followed the benchmark's protocol. Our implementation builds on product quantization with the proposed quantization and SIMD based ADC (Guo et al., 2016b) for distance computation. This is further combined with a vector quantization based tree (Wu et al., 2017). Our implementation is open-source and available at `https://github.com/google-research/google-research/tree/master/scann` and furthermore the exact configurations used to produce our benchmark numbers are part of the ANN-Benchmarks GitHub repository. Figure 4b shows our performance on `Glove1.2M` significantly outperforms competing methods in the high-recall region.

6. Conclusion

In this paper, we propose a new quantization loss function for inner product search, which replaces traditional reconstruction error. The new loss function is weighted based on the inner product values, giving more weight to the pairs of query and database points with higher inner product values. The proposed loss function is theoretically proven and can be applied to a wide range of quantization methods, for example product and binary quantization. Our experiments show superior performance on retrieval recall and inner product value estimation compared to methods that use reconstruction error. The speed-recall benchmark on public datasets further indicates that the proposed method outperforms state-of-the-art baselines which are known to be hard to beat.

References

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 2019.

Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Computer Vision*

and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 931–938. IEEE, 2014.

Artem Babenko, Relja Arandjelović, and Victor Lempitsky. Pairwise quantization. *arXiv preprint arXiv:1606.01550*, 2016.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738, 2015.

Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 557–566, 2015.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. *SPTAG: A library for fast approximate nearest neighbor search*, 2018. URL https://github.com/Microsoft/SPTAG.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 39–46, 2010.

Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 913–922. JMLR. org, 2017.

Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2008.

Thomas Dean, Mark Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine: Technical supplement. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, April 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.240.

Dan Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, 2019.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, 2016a.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pages 482–490, 2016b. URL http://jmlr.org/proceedings/papers/v51/guo16a.html.

Ben Harwood and Tom Drummond. FANNG: Fast approximate nearest neighbour graphs. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 5713–5722. IEEE, 2016.

Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1): 117–128, 2011.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Xiaoyun Li and Ping Li. Random projections with asymmetric quantization. In *Advances in Neural Information Processing Systems*, pages 10857–10866, 2019.

Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Yury A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016. URL http://arxiv.org/abs/1603.09320.

E. Marcheret, V. Goel, and P. A. Olsen. Optimal quantization and bit allocation for compressing large discriminative feature space transforms. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 64–69, Nov 2009.

Julieta Martinez, Shobhit Zakhmi, Holger H Hoos, and James J Little. Lsq++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 491–506, 2018.

Avner May, Jian Zhang, Tri Dao, and Christopher Ré. On the downstream performance of compressed word embeddings. In *Advances in neural information processing systems*, pages 11782–11793, 2019.

Stanislav Morozov and Artem Babenko. Unsupervised neural quantization for compressed-domain similarity search. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11): 2227–2240, 2014.

Stephen Mussmann and Stefano Ermon. Learning and inference via maximum inner product search. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2587–2596, 2016.

Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2827–2836, 2017.

Sashank J Reddi, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. Stochastic negative mining for learning with large output spaces. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1940–1949, 2019.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Herv Jgou. Spreading vectors for similarity search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkGuG2R5tm.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

Santosh S Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston. Starspace: Embed all the things! *arXiv preprint arXiv:1709.03856*, 2017.

Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix Yu. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems 30*, pages 5745–5755. 2017.

Ian En-Hsu Yen, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Sanjiv Kumar, and Pradeep Ravikumar. Loss decomposition for fast learning in large output spaces. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5640–5649, 2018.

Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, volume 2, page 3, 2014.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.