Query-Aware Quantization for Maximum Inner Product Search

Jin Zhang[1], **Defu Lian**[1,2*], **Haodi Zhang**[3], **Baoyun Wang**[4], **Enhong Chen**[1,2]
[1] University of Science and Technology of China
[2] State Key Laboratory of Cognitive Intelligence, Hefei, China
[3] Shenzhen University
[4] Hisense
jinzhang21@mail.ustc.edu.cn, {liandefu, cheneh}@ustc.edu.cn,
hdzhang@szu.edu.cn, wangbaoyun@hisense.com

Abstract

Maximum Inner Product Search (MIPS) plays an essential role in many applications ranging from information retrieval, recommender systems to natural language processing. However, exhaustive MIPS is often expensive and impractical when there are a large number of candidate items. The state-of-the-art quantization method of approximated MIPS is product quantization with a score-aware loss, developed by assuming that queries are uniformly distributed in the unit sphere. However, in real-world datasets, the above assumption about queries does not necessarily hold. To this end, we propose a quantization method based on the distribution of queries combined with sampled softmax. Further, we introduce a general framework encompassing the proposed method and multiple quantization methods, and we develop an effective optimization for the proposed general framework. The proposed method is evaluated on three real-world datasets. The experimental results show that it outperforms the state-of-the-art baselines.

Introduction

Maximum Inner Product Search (MIPS) has wide applicability in recommender systems, information retrieval, and natural language processing. For example, the inner product has been widely used in recommender systems to estimate users' preference for items (Li et al. 2017; Xue et al. 2017; Krichene et al. 2018; Ghosh, Mitra, and Lan 2022), in information retrieval to evaluate the relevance between query and response (Luan et al. 2021; Luo et al. 2022), and in natural language processing to estimate the likelihood of the next word given the context (Zhang et al. 2018; Fu et al. 2021), and then return the results via MIPS.

To define the MIPS problem, consider a large set $\boldsymbol{S}$ of collecting candidate items, where $\boldsymbol{S} \subset \mathbb{R}^d$, and a given query point $\boldsymbol{q} \in \mathbb{R}^d$. The goal is to search for $\boldsymbol{x} \in \boldsymbol{S}$ which maximizes (or approximately maximizes) the inner product $\langle \boldsymbol{q}, \boldsymbol{x} \rangle$. Formally, we are interested in efficiently computing

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \boldsymbol{S}} \langle \boldsymbol{q}, \boldsymbol{x} \rangle .$$

Exhaustive MIPS is often expensive and even impractical when the cardinality of the set is large. Several techniques have been proposed to solve the approximate MIPS problem efficiently, in which the quantization-based techniques have shown strong performance. In (Guo et al. 2020), they developed a score-aware loss on assuming queries with uniform spherical distributions, leading to the state-of-the-art vector quantization that more greatly penalizes the parallel component of a datapoint's residual relative to its orthogonal component. The score-aware loss with the above query assumption is called anisotropic quantization loss, which is considered more suitable for maximum inner product search. However, in real-world datasets, the distribution of queries does not necessarily satisfy the assumption. It will lead to significant differences in the performance on different datasets. Furthermore, we will show that it is not reasonable to penalize the parallel direction more when the distribution of queries does not satisfy the assumption, which is not helpful for a more accurate inner product calculation in the quantization methods.

Motivated by these shortcomings, in this paper, we propose Query-aware Quantization based on the sampled softmax by drawing samples from actual query distribution. Further, we introduce a general framework encompassing multiple quantization methods, including PQ, QUIP, ScaNN, and Query-aware Quantization, elaborate on the insight of these methods, and develop an effective optimization method for the proposed general framework. The proposed method is evaluated on three real-world datasets. The experimental results show that it outperforms the state-of-the-art baselines.

To summarize, we deliver the following contributions:

- To improve the shortage of existing quantization methods, we propose Query-aware Quantization to promote the advancement of approximate MIPS.
- We introduce a general framework encompassing multiple quantization methods, elaborate on the insight of these methods, and develop an effective optimization method for the proposed general framework.
- The proposed quantization method is evaluated on three real-world datasets, where the results demonstrate that our method outperforms the state-of-the-art baselines.

Related Works

In this section, we briefly review the related works about MIPS and quantization. The most related works with us will

*Corresponding author

be described in detail in the next section as preliminaries.

MIPS. The MIPS problem has been studied for more than a decade. In addition to quantization methods, there is a large body of work on graph search, tree search, and hashing. The graph methods (Malkov and Yashunin 2018; Harwood and Drummond 2016) usually use the inner product as a similarity function to construct a graph and then greedily select the neighbor with the highest inner product when searching. The tree search methods (Muja and Lowe 2014; Ram and Gray 2012) usually partition the space recursively, forming a tree and finding the relevant leaves for a given query. The hashing methods (Shrivastava and Li 2014; Neyshabur and Srebro 2015; Andoni et al. 2015; Chen et al. 2019; Ma et al. 2021) usually partition the space using a similarity-preserving hash function, find the relevant buckets for a given query and score only the items in these buckets.

Quantization. Product quantization (Jegou, Douze, and Schmid 2010), as one of the most representative works for quantization, decomposed the vector representation space into the Cartesian product of subspaces. Optimized product quantization (Ge et al. 2013) jointly learned space decomposition and subspace quantization. Composite Quantization (Zhang, Du, and Wang 2014) and Additive Quantization (Babenko and Lempitsky 2014) do not decompose space but directly iteratively learn multiple codebooks. A large body of work (Guo et al. 2016; Johnson, Douze, and Jégou 2019; May et al. 2019; Dai et al. 2020; Guo et al. 2020) is dedicated to improving quantization technology to solve MIPS. Some of them that are highly relevant to us will be covered in detail in next section as preliminary knowledge. Our work is a further improvement of quantization to make it more suitable for maximum inner product search by applying the proposed Query-aware loss.

Preliminaries

Quantization-based methods usually derive multiple codebooks by minimizing the loss function between datapoints and the composition of codewords. It is possible to composite these codewords by concatenation or addition, such that an exponentially large codebook can be generated. The key to quantization methods lies in obtaining codebooks and using them to encode data points. Once we have the codebooks and the codes of data points, we can set up a lookup table (i.e., inner product between query and codebooks) to quickly calculate the approximate inner product between query and data points.

Product Quantization

In product quantization (Jegou, Douze, and Schmid 2010), we split each datapoint $\boldsymbol{x} \in \mathbb{R}^d$ into M subspaces each of dimension d/M, $\boldsymbol{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)}\right)$. We then create M codebooks $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, \ldots, \boldsymbol{C}^{(M)}$, each with K codewords of dimension d/M. Each data point would then be encoded with M dimensions, with every dimension taking one of K states. Moreover, each datapoint $\boldsymbol{x}$ is compressed as $\widetilde{\boldsymbol{x}}$,

$$\widetilde{\boldsymbol{x}} = \left(\boldsymbol{C}^{(1)}_{i_1(\boldsymbol{x})}, \boldsymbol{C}^{(2)}_{i_2(\boldsymbol{x})}, \ldots, \boldsymbol{C}^{(M)}_{i_M(\boldsymbol{x})}\right),$$

where $i_k(\boldsymbol{x}) \in \{1, \ldots, K\}$ returns the index in the k-th codebook of the datapoint $\boldsymbol{x}$, and $\boldsymbol{i}(\boldsymbol{x}) = [i_1(\boldsymbol{x}), i_2(\boldsymbol{x}), \cdots, i_M(\boldsymbol{x})]$ is called code of $\boldsymbol{x}$.

The objective function of product quantization is reconstruction error. Concretely,

$$\ell = \sum_{i=1}^{n} \|\boldsymbol{x_i} - \widetilde{\boldsymbol{x}}_{\boldsymbol{i}}\|^2. \tag{1}$$

Minimizing the error in Eq.(1) is equivalent to solving k-means problems in all subspaces.

Quantization-based MIPS

To apply quantization methods to MIPS, it is more effective to consider the quantization errors caused by the inner product between query and datapoint before and after quantizing than the traditional reconstruction loss.

We start considering the quantization objective of minimizing the expected total inner product quantization errors over the query distribution:

$$\mathbb{E}_{\boldsymbol{q}} \sum_{i=1}^{n} \left(\langle \boldsymbol{q}, \boldsymbol{x_i} \rangle - \langle \boldsymbol{q}, \tilde{\boldsymbol{x}}_{\boldsymbol{i}} \rangle\right)^2 = \mathbb{E}_{\boldsymbol{q}} \sum_{i=1}^{n} \langle \boldsymbol{q}, \boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}} \rangle^2 . \tag{2}$$

It takes expectation over all possible combinations of datapoints $\boldsymbol{x}$ and queries $\boldsymbol{q}$, where $\tilde{\boldsymbol{x}}$ is the compressed vector of $\boldsymbol{x}$ by quantization.

In QUIP (Guo et al. 2016), they consider Eq.(2) in each subspace with the assumption that each subspace is independent of others i.e., the dimensions of different subspaces are not related. Meanwhile, given a held-out set of samples $\boldsymbol{Z}$, which is sampled from the same distribution as the query, in the k-th subspace, they consider the following loss :

$$\ell^{(k)} = \sum_{i=1}^{n} \left(\boldsymbol{x_i}^{(k)} - \boldsymbol{C}^{(k)}_{i_k(\boldsymbol{x_i})}\right)^{\top} \boldsymbol{\Sigma}^{(k)}_{\boldsymbol{Z}} \left(\boldsymbol{x_i}^{(k)} - \boldsymbol{C}^{(k)}_{i_k(\boldsymbol{x_i})}\right), \tag{3}$$

where $\boldsymbol{\Sigma}^{(k)}_{\boldsymbol{Z}} = \frac{1}{|\boldsymbol{Z}|} \sum_{\boldsymbol{z} \in \boldsymbol{Z}} \boldsymbol{z}^{(k)} \boldsymbol{z}^{(k)\top}$ is the non-centered covariance matrix in subspace k estimated from held-out sample set $\boldsymbol{Z}$. Minimizing the error in Eq.(3) is equivalent to solving a modified k-means problem in the k-th subspace.

In ScaNN (Guo et al. 2020), they also start from objective function Eq.(2), and the core idea is that not all pairs of $(\boldsymbol{x}, \boldsymbol{q})$ are equally important. The approximation error on the pairs with a high inner product is far more important since they are likely among the top-ranked pairs and can greatly affect the search result. Concretely, they proposed score-aware loss :

$$\ell(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}}, w) = E_{\boldsymbol{q} \sim Q}[w(\langle \boldsymbol{q}, \boldsymbol{x_i} \rangle) \langle \boldsymbol{q}, \boldsymbol{x_i} - \widetilde{\boldsymbol{x}}_{\boldsymbol{i}} \rangle^2], \tag{4}$$

where $w : \mathbb{R} \to \mathbb{R}^+$ is a weight function of the inner product score. They developed score-aware loss on assuming queries with uniform spherical distributions. Meanwhile, they use indicator function $w(t) = \mathbb{I}(t > T)$ as weight function leading to anisotropic quantization loss. By defining the residual error $\boldsymbol{r_i} = \boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i$, the parallel residual error which is the

component of the residual error parallel to the datapoint $\boldsymbol{x_i}$, and orthogonal residual error computed as

$$\boldsymbol{r}_{\parallel} = \frac{\langle \boldsymbol{r_i}, \boldsymbol{x}_i \rangle \, \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|^2}, \quad \boldsymbol{r}_{\perp} = \boldsymbol{r_i} - \boldsymbol{r}_{\parallel},$$

they have shown that the anisotropic quantization loss can be computed as

$$\ell(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}}) = h_{i,\parallel} \|\boldsymbol{r}_{\parallel}\|^2 + h_{i,\perp} \|\boldsymbol{r}_{\perp}\|^2, \tag{5}$$

where $h_{i,\parallel}$ and $h_{i,\perp}$ are constant with respect to $\|\boldsymbol{x_i}\|$ and T.

General Framework

In this section, we introduce a general framework encompassing existing quantization methods, including PQ, QUIP, and ScaNN. Then we use the orthogonal decomposition of matrices to give the geometric intuition for these methods as insights, based on which we propose the improved method called Query-aware Quantization. Finally, we develop an effective optimization method for the proposed general framework and analyze its complexity and convergence.

General Framework

We show the following property for anisotropic quantization loss to unify the multiple methods into a unified form.

Proposition 1. *By defining*

$$\boldsymbol{M_i} = \frac{\left(h_{i,\parallel} - h_{i,\perp}\right) \boldsymbol{x_i} \boldsymbol{x_i}^{\top}}{\|\boldsymbol{x_i}\|^2} + h_{i,\perp} \boldsymbol{I_d},$$

where $\boldsymbol{I_d}$ is the d-dimensional identity matrix, Eq.(5) *will be equal to*

$$\ell(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}}) = \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}}\right)^{\top} \boldsymbol{M_i} \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}}\right). \tag{6}$$

The proof is attached in the Appendix.9.

Combining Eqs.(1)(3)(6), it can be seen that they all have a uniform form:

$$\ell = \sum_{i=1}^{n} \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}}\right)^{\top} \boldsymbol{M_i} \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}}\right). \tag{7}$$

The matrix $\boldsymbol{M_i}$ in different quantization methods is summarized in Table 1. We can see that the essence of applying the quantization methods in the MIPS problem is to replace the traditional l_2 norm in the objective function with the norm induced by the positive definite matrix $\boldsymbol{M_i}$. The key is what kind of matrix is suitable. To answer the question, we need to have a deeper understanding of this type of loss function.

Insights

When we employ the quantization method, it is unavoidable that there is always an error between the compressed data points $\tilde{\boldsymbol{x}}$ and the original data points $\boldsymbol{x}$. In the calculation of the inner product, we know that not every direction is equally important. For example, when we calculate $\langle \boldsymbol{q}, \boldsymbol{x} \rangle$, the direction of $\boldsymbol{q}$ is the most important, and if the residual vector $\boldsymbol{r} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$ has a small component in the direction where $\boldsymbol{q}$ is located, then the approximate inner product

PQ	$\forall i, \boldsymbol{M_i} = \boldsymbol{I}$
QUIP	$\forall i, \boldsymbol{M_i} = \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(1)} \bigoplus \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(2)} \bigoplus \ldots \bigoplus \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(M)}$
ScaNN	$\boldsymbol{M_i} = \frac{\left(h_{i,\parallel} - h_{i,\perp}\right) \boldsymbol{x_i} \boldsymbol{x_i}^{\top}}{\|\boldsymbol{x_i}\|^2} + h_{i,\perp} \boldsymbol{I}$
Ours	$\boldsymbol{M_i} = \sum_{\boldsymbol{q} \in \boldsymbol{Z}} p(\boldsymbol{q}\|\boldsymbol{x_i}) \boldsymbol{q} \boldsymbol{q}^{\top}$

Table 1: $\boldsymbol{M_i}$ in different quantization methods.

$\langle \boldsymbol{q}, \tilde{\boldsymbol{x}} \rangle$ will be calculated more accurately. If the quantization is accurate in some important directions, the approximate inner product will be computed more accurately. We will illustrate that the positive definite matrix $\boldsymbol{M}$ is essentially an adaptive assignment of importance to different directions.

We start with the orthogonal decomposition of $\boldsymbol{M}$. We can obtain

$$\boldsymbol{M} = \boldsymbol{V}^{\top} \boldsymbol{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_d) \boldsymbol{V},$$

where $\boldsymbol{V}$ is orthogonal matrix, λ_i is eigenvalue of $\boldsymbol{M}$, without loss of generality, assuming

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0.$$

Through orthogonal matrix, we can obtain d orthogonal unit vectors as a new basis, $\boldsymbol{V}^{\top} = (\boldsymbol{v_1}, \cdots, \boldsymbol{v_d})$. We regain the representation of residual vector $\boldsymbol{r}$ with $\boldsymbol{V}$ as the basis

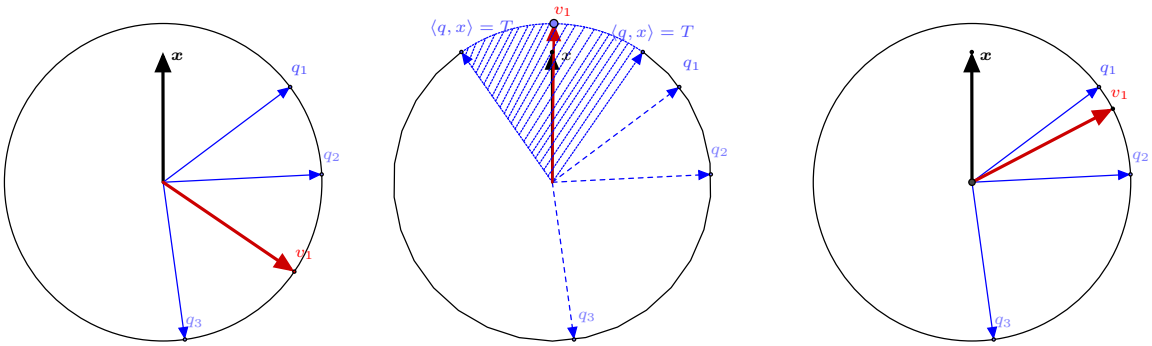
$$\boldsymbol{r} = \sum_{k=1}^{d} \boldsymbol{r}_k = \sum_{k=1}^{d} \langle \boldsymbol{r}, \boldsymbol{v}_k \rangle \, \boldsymbol{v}_k.$$

Then we go back to the loss and obtain

$$\begin{aligned} \ell &= \left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right)^{\top} \boldsymbol{M} \left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right) \\ &= \boldsymbol{r}^{\top} \boldsymbol{V}^{\top} \boldsymbol{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_d) \boldsymbol{V} \boldsymbol{r} \\ &= \sum_{k=1}^{d} \lambda_k \|\boldsymbol{r}_k\|^2. \end{aligned} \tag{8}$$

From Eq.(8), we can see that the direction $\boldsymbol{r_1}$, parallel to $\boldsymbol{v_1}$, is most important because it has the largest weight in the loss function, which means that the quantization in direction $\boldsymbol{v}_1$ is the most accurate than the other orthogonal directions. This means that if the direction of a query is similar to the $\boldsymbol{v_1}$, the inner product between query and datapoint will be calculated more accurately compared with the other queries.

Let us review existing works. The traditional approach PQ considers each direction equally important, which does not help in the MIPS task. Quantization-based MIPS will consider different importance of directions. In Fig.1, we assume that there are some available queries and show the most important directions $\boldsymbol{v_1}$ in the different methods. In the QUIP, the most important direction $\boldsymbol{v_1}$ is the mean direction of all available queries. In the ScaNN, for single point $\boldsymbol{x}$, they don't use actual queries but assume queries uniformly distribute in the blue circumferential part of Fig.1b, and the most important direction is parallel to $\boldsymbol{x}$. This again shows that ScaNN penalizes the parallel direction (i.e., parallel to $\boldsymbol{x}$) more, which is consistent with their conclusion.



(a) QUIP (b) ScaNN (c) Ours

Figure 1: Insights into different loss functions.

In Fig.1, we can see that the inner product $\langle \boldsymbol{q_1}, \boldsymbol{x} \rangle$ is maximum. For the goal of MIPS, we hope to estimate it as accurately as possible, which means that it is ideal for $\boldsymbol{v_1}$ to be close to $\boldsymbol{q_1}$, not to $\boldsymbol{x}$. Based on such observations, we proposed Query-aware Quantization in the next subsection.

Query-aware Quantization

In this subsection, we propose our method called Query-aware Quantization. The proposed method aims to approximate the inner product with high values more accurately. To achieve this, we assign higher weights to higher inner product pairs. A common choice among weighting methods is to use an exponential function. Empirically, we found that using softmax gives better results.

We introduce proposed Query-aware loss based on the sampled softmax, by drawing samples from actual query distributions. The formal loss function is defined as follows:

Definition 1. *Given a datapoint $\boldsymbol{x_i}$, its quantization $\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}$, the Query-aware quantization loss with respect to samples $\boldsymbol{Z}$ which is sampled from the same distribution as the query is defined as*

$$\ell_i = \sum_{\boldsymbol{q} \in \boldsymbol{Z}} p(\boldsymbol{q}|\boldsymbol{x_i}) \left(\langle \boldsymbol{q}, \boldsymbol{x_i} \rangle - \langle \boldsymbol{q}, \tilde{\boldsymbol{x}}_{\boldsymbol{i}} \rangle \right)^2, \tag{9}$$

where $p(\boldsymbol{q}|\boldsymbol{x_i})$ is output of softmax in $\{\langle \boldsymbol{q}, \boldsymbol{x_i} \rangle : \boldsymbol{q} \in \boldsymbol{Z}\}$.

By defining

$$\boldsymbol{M_i} = \sum_{\boldsymbol{q} \in \boldsymbol{Z}} p(\boldsymbol{q}|\boldsymbol{x_i}) \boldsymbol{q}\boldsymbol{q}^\top, \tag{10}$$

named as Query-aware Matrix, Eq.(9) is equal to

$$\ell_i = \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}} \right)^\top \boldsymbol{M_i} \left(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}} \right). \tag{11}$$

This shows that the proposed approach also belongs to the generalized framework summarized in Tab.1. The most important quantization direction is shown in Fig.1c, which is closer to $\boldsymbol{v_1}$ than other methods.

In practice, although the above loss function has nice geometric intuition, it is computationally difficult. To calculate above loss efficiently, one naive method is to store $\boldsymbol{M_i}$ for each datapoint $\boldsymbol{x_i}$. Although it can speed up the computation of the loss function, it requires a large amount of memory, which is quite expensive and does not work in large datasets. We combine local shared and sampled softmax to solve this problem.

Local Shared. Previous work (Guo et al. 2020) also had the above problem, and their approach was that all data points share the same ratio(i.e., $h_{i,\parallel}/h_{i,\perp}$) computed in the sense of the limit. Inspired by this, we use a clustering algorithm to group the data points, with each group of data points shared the same Query-aware Matrix. Empirically, this approach performs well when the number of clustering centroids is relatively large.

Sampled Softmax. The $\boldsymbol{M_i}$ in Eq.(10) is in essence the expectation:

$$\boldsymbol{M_i} = \mathbb{E}_{\boldsymbol{q} \sim P_i} \left[\boldsymbol{q}\boldsymbol{q}^\top \right], \tag{12}$$

where P_i denotes $p(\boldsymbol{q}|\boldsymbol{x_i})$. Similar to (Jean et al. 2014), to approximate this expectation, we sample a small subset uniformly from the large held-out set identically distributed as the query. This approach allows us to compute the $\boldsymbol{M_i}$ using only a small subset, decreasing computational complexity.

Using our Query-aware loss function Eq.(11), we obtain a new objective function for product quantization we call the Query-aware product quantization problem.

Definition 2. *Given a dataset $x_1, x_2, \ldots, x_n$ of points in $\mathbb{R}^d$, a number M of codebooks each with elements of size d/M and k codewords in each codebook, the Query-aware product quantization problem is to find the M codebooks that minimizes*

$$\min_{C^{(1)}, \ldots, C^{(M)}} \sum_{i=1}^{n} \min_{\tilde{\boldsymbol{x}}_i \in \Pi_{m=1}^{M} C^{(m)}} \ell_i, \tag{13}$$

where ℓ_i is defined in Eq.(11).

Algorithm 1: Iterative optimization

Input: Train set $\boldsymbol{X_t}$, samples $\boldsymbol{Z}$, the codebook $\boldsymbol{c}$
Output: The codebook $\boldsymbol{c}$

for *iter* $\leftarrow 1$ **to** *max_iter* **do**
  $\boldsymbol{b} \leftarrow$ update indices as described in Eq.(16) ;
  $\boldsymbol{c} \leftarrow$ update codebooks with $\boldsymbol{Z}$ by Eq.(17);
return $\boldsymbol{c}$;

Algorithm 2: Query-aware Quantization

Input: Database $\boldsymbol{X}$, validation set $\boldsymbol{V}$, held-out set $\boldsymbol{H}$, integer N, k_v.
Output: The codebook and codes of Database.

$\boldsymbol{c} \leftarrow$ initialize codebook by selecting K points in $\boldsymbol{X}$ randomly and concatenate it in order with respect to subspace ;
$\boldsymbol{X_c} \leftarrow$ clustering algorithm to obtain k_v centroids;
for *iter* $\leftarrow 1$ **to** *#iter* **do**
  $\boldsymbol{Z} \leftarrow$ sample N datapoints uniformly from $\boldsymbol{H}$;
  $\boldsymbol{M_i} \leftarrow$ compute Quary-aware Matrix w.r.t. $\boldsymbol{X_c}$;
  $\boldsymbol{c} \leftarrow$ optimize codebook by Alg.1;
  $\boldsymbol{Z_{best}}, \boldsymbol{c_{best}} \leftarrow$ evaluate on $\boldsymbol{V}$, save best samples and best codebooks according to evaluation metric;
$codes \leftarrow$ use $\boldsymbol{Z_{best}}, \boldsymbol{c_{best}}$ to encode database $\boldsymbol{X}$ as described in Eq.(16).
return $\boldsymbol{c}, codes$;

We will discuss the optimization method for the above problem in the following subsection. The complete process of Query-aware Quantization is described in Alg.2. The complexity and convergence will be analyzed latter.

Optimization Procedure

To explicitly formulate the objective loss Eq.(7), we can represent each index $i_m(\boldsymbol{x_i})$ as a one-hot vector and concatenate these one-hot vectors in a specified order to obtain the index matrix $\boldsymbol{b}_i \in \{0,1\}^{M \times MK}$,

$$\boldsymbol{b}_i = \begin{bmatrix} \boldsymbol{b}_i^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{b}_i^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{b}_i^{(M)} \end{bmatrix}$$

where the k-th element $b_{i,k}^{(m)}$ of the K-dimensional vector $\boldsymbol{b}_i^{(m)}$ equals to 1 if $i_m(\boldsymbol{x}_i) = k$ and 0 otherwise. If we further concatenate the codebooks $\boldsymbol{C}^{(m)}$ to obtain a column vector

$$\boldsymbol{c} = \left[\boldsymbol{C}_1^{(1)}, \boldsymbol{C}_2^{(1)}, \cdots, \boldsymbol{C}_K^{(1)}, \boldsymbol{C}_1^{(2)}, \cdots, \boldsymbol{C}_K^{(M)}\right]^\top \in \mathbb{R}^{Kd \times 1},$$

by setting $\boldsymbol{B_i} = \boldsymbol{b}_i \otimes \boldsymbol{I}_{d/M}$, then

$$\widetilde{\boldsymbol{x}} = \boldsymbol{B_i c}, \tag{14}$$

where $\otimes$ denotes Kronecker Product, $\boldsymbol{I}_{d/M}$ is the d/M-dimensional identity matrix. Combining Eq.(7) and Eq.(14), the objective loss function is equal to

$$\ell(\boldsymbol{c}, \boldsymbol{b}) = \sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{B_i c})^\top \boldsymbol{M_i} (\boldsymbol{x_i} - \boldsymbol{B_i c}). \tag{15}$$

Next, we develop an iterative algorithm to optimize the above loss. Like most quantization works (Ge et al. 2013; Zhang, Du, and Wang 2014; Guo et al. 2020), our algorithm iterates between minimization over the index $\boldsymbol{b}$ update step and codebooks $\boldsymbol{c}$ update step. The process is described in Alg.1.

Update$\{\boldsymbol{b}_i\}$. It can be easily seen that $\boldsymbol{b}_i$, the index matrix of $\boldsymbol{x}_i$, given $\boldsymbol{c}$ fixed, is independent to $\{\boldsymbol{b}_j\}_{j \neq i}$, the optimization problem is decomposed to n subproblems. To obtain the index $\boldsymbol{b}_i$, we need to optimize

$$\boldsymbol{b}_i^* = \arg\min_{\boldsymbol{b}_i} \ell_i(\boldsymbol{c}, \boldsymbol{b_i}). \tag{16}$$

This involves a combinatorial optimization problem, which suffers from high computational costs. Similar to (Zhang, Du, and Wang 2014), we use the alternative optimization technique to optimize it, and solve the subvectors $\{\boldsymbol{b}_i^{(m)}\}_{m=1}^M$ alternatively. Concretely, given $\{\boldsymbol{b}_i^{(l)}\}_{l \neq m}$ fixed, we update $\boldsymbol{b}_i^{(m)}$ by exhaustively checking all the elements in the codebook $\boldsymbol{C}^{(m)}$, finding the element such that the objective value is minimized, and accordingly setting the corresponding entry of $\boldsymbol{b}_i^{(m)}$ to be 1 and all the others to be 0.

Update$\{\boldsymbol{c}\}$. Fixing $\boldsymbol{b}$, the codebook $\boldsymbol{c}$ is learned with minimizing the Eq.(15),

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \ell(\boldsymbol{c}, \boldsymbol{b}).$$

This is a convex quadratic minimization problem. By setting

$$\boldsymbol{T} = \sum_i \boldsymbol{B}_i^\top \boldsymbol{M_i B}_i, \quad \boldsymbol{r} = \sum_i \boldsymbol{B}_i^\top \boldsymbol{M_i x}_i,$$

the closed-form solution can be given as follows:

$$\boldsymbol{c} = \boldsymbol{T}^{-1} \boldsymbol{r}. \tag{17}$$

$\boldsymbol{T}$ will be full rank if we observe every codeword at least once; otherwise consider adding a regular term.

Complexity and Convergence Analysis

In this subsection, we analyze the complexity and convergence of the proposed algorithms.

Complexity Alg.2 runs Alg.1 multiple times and the computation of Alg.1 mainly comes from updating indices and codebooks.

In terms of updating indices, the encoding time complexity of each data point is $\mathcal{O}(I_e MKd^2 + Nd^2)$. The former part is related to exhaustively checking all the elements in every codebook, and the latter part is due to matrix computation related to $\boldsymbol{M_i}$, where I_e is the maximum number of iteration rounds in updating $\boldsymbol{b}_i$. In our experiments, I_e is set to 3.

In terms of updating codebooks, we need to compute $\boldsymbol{T}$ and $\boldsymbol{r}$. It is worth noting that we can avoid high-dimensional

matrix multiplication by virtue of the sparsity structure of matrix $\boldsymbol{B_i}$. Concretely, we extract the non-zero column index of the matrix $\boldsymbol{B_i}$ as $\boldsymbol{\beta_i}$. The length of $\boldsymbol{\beta_i}$ is equal to d. The calculation formulas are as follows:

$$\boldsymbol{T}[\boldsymbol{\beta_i} \times \boldsymbol{\beta_i}] \leftarrow \boldsymbol{T}[\boldsymbol{\beta_i} \times \boldsymbol{\beta_i}] + \boldsymbol{M_i},$$
$$\boldsymbol{r}[\boldsymbol{\beta_i}] \leftarrow \boldsymbol{r}[\boldsymbol{\beta_i}] + \boldsymbol{M_i}\boldsymbol{x_i},$$

where $\times$ is the Cartesian product. The complete update of codebooks requires $\mathcal{O}(nd^2 + K^3d^3)$, the former part is matrix computation related to $\boldsymbol{T}$ and $\boldsymbol{r}$, the latter part is due to matrix inverse $\boldsymbol{T}^{-1}$. The computation of $\boldsymbol{M_i}$ is completed in updating indices. In our experiments, K is set to 16.

Convergence In this part, we discuss the convergence of the proposed algorithm. The objective loss in each iteration is the same if the expectation Eq.(12) is reasonably approximated. In addition, the essence of the iterative algorithm Alg.1 is the coordinate descent method, which converges to a local minimum. That is, the convergence of the Alg.2 is certain as long as the expectation value is reasonably approximated. To achieve this, in our experiments, we set the number of samples N to 500. Concerning some bad samples, we set max_iter to 2 and save the best codebooks and samples during training.

Experiments

In this section, we conduct experiments on three real-world datasets to show our method leads to improved performance on MIPS. First, we will introduce the three datasets used for the experimental evaluation. Then, the direct comparison of the relative loss under multiple codebook configurations shows that our method is more accurate in approximating the high inner product values. Finally, we conducted recall experiments to compare with baselines. Our experiments will show that the proposed method performs well on all these datasets. In contrast, other methods are limited by their specific assumptions and cannot achieve good performance on all datasets. Experimental results on more datasets, and the analysis of various parts of the algorithm, including the effectiveness of the local shared matrix, the selection of the weight function, and the convergence of the algorithm are attached in Appendix. The proposed algorithm is implemented in Julia, and all experiments are conducted in a Linux server with 3.00GHZ intel GPU and 300G main memory.

Datasets

Our method is evaluated on three real datasets, LastFM, Glove, and Music100, which were used in the evaluation of multiple MIPS algorithms (Morozov and Babenko 2018; Guo et al. 2020; Krishnan and Liberty 2021; Zhang et al. 2022). The dataset splitting and experimental results on other datasets, including EchoNest and Yahoo!Music, are attached in the Appendix.

The LastFM dataset is a real music recommendation dataset that includes $357,847$ users and $156,122$ items. We use matrix decomposition to train them into 100-dimensional embedding vectors as described in (Lian et al. 2015). We also conduct experiments with other embedding dimensions, which are attached in the Appendix.

The Glove dataset is a collection of 1.2 million 100-dimensional word embedding trained as described in (Pennington, Socher, and Manning 2014). It is the only dataset used for evaluation in ScaNN because it conforms to their assumptions. It is worth noting that in some of the previous work like ScaNN, the norm of data points in this dataset is normalized, at which point the MIPS task is essentially equivalent to the nearest neighbor search task. We don't do such pre-processing and evaluate on the MIPS task.

The Music100 dataset was introduced in IP-NSW (Morozov and Babenko 2018). This dataset was obtained by IALS factorization (Hu, Koren, and Volinsky 2008) of the user-item ranking matrix, with dimensionality 100. The matrix contains the ratings from $3,897,789$ users on one million popular songs from a proprietary music recommendation service.

Direct Comparison in Relative Error

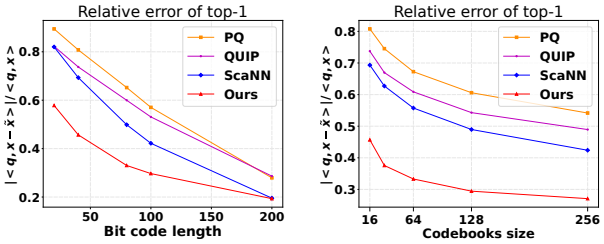


Figure 2: The relative error of inner product estimation for true Top-1 on Music100 across multiple code bit lengths and codebooks size settings.

We start by directly comparing the accuracy of the estimated top-1 inner product as measured by relative error $|\frac{\langle \boldsymbol{q},\boldsymbol{x}-\widetilde{\boldsymbol{x}}\rangle}{\langle \boldsymbol{q},\boldsymbol{x}\rangle}|$, meanwhile, study the effect of different codebook configurations on the performance of the proposed method.

In the left of Fig.2, we fix the number of codewords in each codebook (i.e., $K = 16$) and compare the relative errors under different codebook numbers. In the right of Fig.2, we fix the number of codebooks (i.e., $M = 10$) and compare the relative errors under different codebook sizes. The results show that overall bit settings and codebooks size, our proposed method leads to smaller relative error, especially when a smaller number of codebooks is used.

Comparison with Baselines

Baselines In this subsection, we introduce the baseline methods for comparison. The compared methods are as follows.

- **PQ (Jegou, Douze, and Schmid 2010).** PQ and our method are compared on the same number of codebooks and the same number of codewords.
- **QUIP (Guo et al. 2016).** Our methods compared with QUIP-cov which is described in preliminaries. Both methods use the same held-out samples.
- **ScaNN (Guo et al. 2020).** The anisotropic quantization threshold is tuned to obtain the best result. Concretely, we tune the anisotropic quantization thresholds

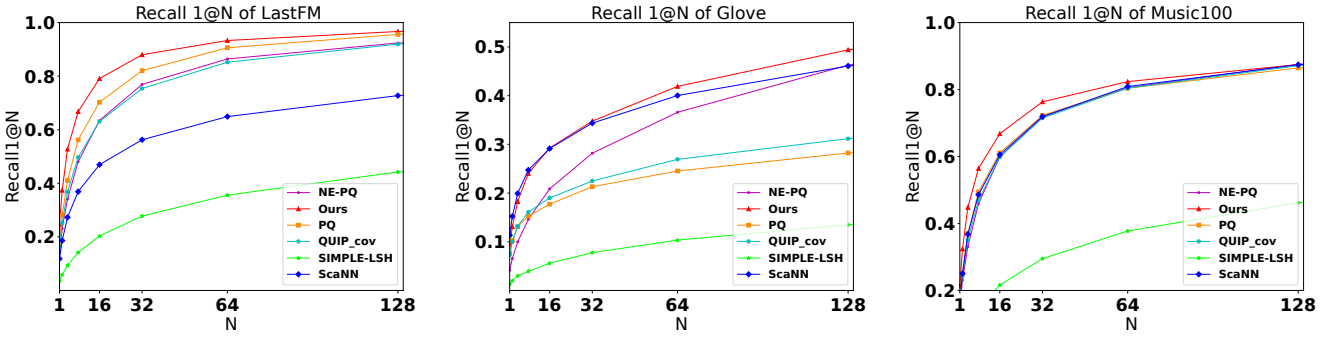


Figure 3: Recall 1@N curve comparing with baselines on MIPS tasks.

in $\{0.01, 0.05, 0.1, 0.2, 0.3\}$ times the mean norm of data points.

- **SIMPLE-LSH (Neyshabur and Srebro 2015).** The hash code length is set to the bits used in quantization methods, which occupies the same storage as quantization-based methods. We return the neighbors based on the hamming distance, as described in the (Neyshabur and Srebro 2015).
- **NEQ (Dai et al. 2020).** Our methods compared with NE-PQ which applies NEQ to PQ. The proposed method and NE-PQ both use the PQ technique.

Settings For all quantization methods, we set the dimension of the subspace to 4, i.e., 25 codebooks, and use 16 codewords in each codebook, which is the same as the settings in ScaNN (Guo et al. 2020). Each datapoint will be quantized to code which takes up 100 bits of storage. Accordingly, the length of the hash code is set to 100. In our method Alg.2, we set N to 500, #iter to 20, max_iter to 2, and k_v to 2000. We use Recall as the evaluation metric, and k-means as clustering algorithm. In the Appendix, we will explain in detail how these hyper-parameters are set.

Evaluation Metrics One of the most important measures of the performance of a MIPS system is Recall. The Recall k@N measure is defined as a probability (computed over a number of queries) that the set of N closest neighbors returned by algorithms contains the actual top-k nearest neighbors. Sometimes this metric is abbreviated as Recall@N when ground-truth (i.e., top-k nearest neighbors) is specified.

Formally, for a set of queries Q, a query $\boldsymbol{q} \in Q$, let $\pi_N(\boldsymbol{q})$ be the top-N points returned by algorithms, $TN_k(\boldsymbol{q})$ is true top-k nearest neighbor of $\boldsymbol{q}$. Then, the Recall k@N for the set of queries Q is the following quantity:

$$\text{Recall k@N} = \frac{1}{|Q|} \sum_{\boldsymbol{q} \in Q} \frac{|TN_k(\boldsymbol{q}) \cap \pi_N(\boldsymbol{q})|}{|TN_k(\boldsymbol{q})|}.$$

We generate ground-truth results using brute force search and compare the neighbors returned by each algorithm against ground-truth. Following the standard quantization method evaluation (Jegou, Douze, and Schmid 2010; Guo et al. 2016; Wu et al. 2017; Dai et al. 2020; Zhang et al. 2022), the results are shown in the form of Recall 1@N-vs-N curves.

Results From the recall curves in Fig.3, it is clear that our method performs better than all baselines across three datasets. Besides, we can see that QUIP and ScaNN perform differently on different datasets. On LastFM, they don't even improve PQ. On Music100, they are close to PQ, but on Glove, ScaNN far outperforms PQ. NEQ, as a quantization method from the perspective of norm and direction, has less improvement effect on PQ than ours.

Speed Benchmarks In terms of quantization methods, we use the same codebook configuration and the same ADC computation implementation for all quantization methods. Thus, all quantization methods' speed is identical at the same bitrate, meaning Fig.3 are both fixed-memory and fixed-time and therefore directly comparable.

In terms of hashing methods, the speed of computing a score (i.e., hamming distance) for a datapoint is related to bits. Following previous work (Dai et al. 2020), the bits setting is the same as the quantization methods. The quantization methods are faster because they only take $O(M)$ to compute the approximate inner product for a datapoint, where the number of codebooks M is less than bits.

Conclusions

In this paper, to promote the advancement of approximate MIPS, we propose the Query-aware Quantization. We introduce a general framework encompassing the proposed method and multiple quantization methods, and develop an effective optimization method for the proposed general framework. We show that the difference between multiple quantization methods is the difference in the weight assigned to the quantization direction, which can be decided by positive definite matrices, and using appropriate positive definite matrices will greatly improve the performance of quantization methods on MIPS. We conduct recall experiments on three real-world datasets. The results show that the proposed method outperforms the state-of-the-art baselines with respect to approximate search accuracy in fixed time.

Acknowledgments

The work was supported by grants from the National Natural Science Foundation of China (No. 61976198 and 62022077).

References

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. *arXiv preprint arXiv:1509.02897*.

Babenko, A.; and Lempitsky, V. 2014. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 931–938.

Chen, L.; Esfandiari, H.; Fu, G.; and Mirrokni, V. 2019. Locality-sensitive hashing for f-divergences: Mutual information loss and beyond. *Advances in Neural Information Processing Systems*, 32: 10044–10054.

Dai, X.; Yan, X.; Ng, K. K.; Liu, J.; and Cheng, J. 2020. Norm-explicit quantization: Improving vector quantization for maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 51–58.

Fu, H.; Zhou, S.; Yang, Q.; Tang, J.; Liu, G.; Liu, K.; and Li, X. 2021. LRC-BERT: latent-representation contrastive knowledge distillation for natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12830–12838.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4): 744–755.

Ghosh, A.; Mitra, S.; and Lan, A. 2022. DiPS: Differentiable Policy for Sketching in Recommender Systems. *AAAI Conference on Artificial Intelligence*, 6703–6712.

Guo, R.; Kumar, S.; Choromanski, K.; and Simcha, D. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, 482–490. PMLR.

Guo, R.; Sun, P.; Lindgren, E.; Geng, Q.; Simcha, D.; Chern, F.; and Kumar, S. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 3887–3896. PMLR.

Harwood, B.; and Drummond, T. 2016. Fanng: Fast approximate nearest neighbour graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5713–5722.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, 263–272. Ieee.

Jean, S.; Cho, K.; Memisevic, R.; and Bengio, Y. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Jegou, H.; Douze, M.; and Schmid, C. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1): 117–128.

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3): 535–547.

Krichene, W.; Mayoraz, N.; Rendle, S.; Zhang, L.; Yi, X.; Hong, L.; Chi, E.; and Anderson, J. 2018. Efficient training on very large corpora via gramian estimation. *arXiv preprint arXiv:1807.07187*.

Krishnan, A.; and Liberty, E. 2021. Projective Clustering Product Quantization. *arXiv preprint arXiv:2112.02179*.

Li, H.; Chan, T. N.; Yiu, M. L.; and Mamoulis, N. 2017. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, 835–850.

Lian, D.; Ge, Y.; Zhang, F.; Yuan, N. J.; Xie, X.; Zhou, T.; and Rui, Y. 2015. Content-aware collaborative filtering for location recommendation based on human mobility data. In *2015 IEEE international conference on data mining*, 261–270. IEEE.

Luan, Y.; Eisenstein, J.; Toutanova, K.; and Collins, M. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics*, 9: 329–345.

Luo, M.; Mitra, A.; Gokhale, T.; and Baral, C. 2022. Improving Biomedical Information Retrieval with Neural Retrievers. *AAAI Conference on Artificial Intelligence*, 11038–11046.

Ma, C.; Yu, F.; Yu, Y.; and Li, W. 2021. Learning Sparse Binary Code for Maximum Inner Product Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3308–3312.

Malkov, Y. A.; and Yashunin, D. A. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4): 824–836.

May, A.; Zhang, J.; Dao, T.; and Ré, C. 2019. On the downstream performance of compressed word embeddings. *Advances in neural information processing systems*, 32: 11782.

Morozov, S.; and Babenko, A. 2018. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems*, 31: 4721–4730.

Muja, M.; and Lowe, D. G. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11): 2227–2240.

Neyshabur, B.; and Srebro, N. 2015. On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning*, 1926–1934. PMLR.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Ram, P.; and Gray, A. G. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 931–939.

Shrivastava, A.; and Li, P. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *arXiv preprint arXiv:1405.5869*.

Wu, X.; Guo, R.; Suresh, A. T.; Kumar, S.; Holtmann-Rice, D. N.; Simcha, D.; and Yu, F. 2017. Multiscale quantization for fast similarity search. *Advances in neural information processing systems*, 30.

Xue, H.-J.; Dai, X.; Zhang, J.; Huang, S.; and Chen, J. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*, volume 17, 3203–3209. Melbourne, Australia.

Zhang, J.; Liu, Q.; Lian, D.; Liu, Z.; Wu, L.; and Chen, E. 2022. Anisotropic Additive Quantization for Fast Inner Product Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 4354–4362.

Zhang, M.; Liu, X.; Wang, W.; Gao, J.; and He, Y. 2018. Navigating with graph representations for fast and scalable decoding of neural language models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6311–6322.

Zhang, T.; Du, C.; and Wang, J. 2014. Composite quantization for approximate nearest neighbor search. In *International Conference on Machine Learning*, 838–846. PMLR.