Anisotropic Additive Quantization for Fast Inner Product Search

Jin Zhang[1], **Qi Liu**[1], **Defu Lian**[1*], **Zheng Liu**[2], **Le Wu**[3], **Enhong Chen**[1]

[1] University of Science and Technology of China
[2]Microsoft Research Asia [3]Hefei University of Technology
{abczj, qiliu67}@mail.ustc.edu.cn, {liandefu,cheneh}@ustc.edu.cn
Zheng.Liu@microsoft.com, lewu.ustc@gmail.com

Abstract

Maximum Inner Product Search (MIPS) plays an important role in many applications ranging from information retrieval, recommender systems to natural language processing and machine learning. However, exhaustive MIPS is often expensive and impractical when there are a large number of candidate items. The state-of-the-art approximated MIPS is product quantization with a score-aware loss, which weighs more heavily on items with larger inner product scores. However, it is challenging to extend the score-aware loss for additive quantization due to parallel-orthogonal decomposition of residual error. Learning additive quantization with respect to this loss is important since additive quantization can achieve a lower approximation error than product quantization. To this end, we propose a quantization method called Anisotropic Additive Quantization to combine the score-aware anisotropic loss and additive quantization. To efficiently update the codebooks in this algorithm, we develop a new alternating optimization algorithm. The proposed algorithm is extensively evaluated on three real-world datasets. The experimental results show that it outperforms the state-of-the-art baselines with respect to approximate search accuracy while guaranteeing a similar retrieval efficiency.

1 Introduction

Maximum Inner Product Search (MIPS) has wide applicability in recommender systems, information retrieval, natural language processing and machine learning. For example, the inner product has been widely used in recommender systems to estimate users' preference for items (Krichene et al. 2018; Li et al. 2017; Xue et al. 2017; Koren, Bell, and Volinsky 2009; Cremonesi, Koren, and Turrin 2010), in information retrieval to estimate the relevance between query and response (Weston, Bengio, and Usunier 2010; Luan et al. 2021) and in natural language process to estimate the likelihood of the next word given the context (Vaswani et al. 2017; Zhang et al. 2018). Lately, MIPS has also been applied for 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).

[*]Corresponding Author

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

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

Exhaustively computing the exact inner product between query and the set $\boldsymbol{S}$ is often expensive and even impractical when the cardinality of the set is large. Several techniques have been proposed in the literature including hashing (Chen et al. 2019; Neyshabur and Srebro 2015; Shrivastava and Li 2014), graph-based search (Liu et al. 2020; Morozov and Babenko 2018), or quantization (Guo et al. 2020, 2016; Babenko and Lempitsky 2014; Ge et al. 2013) to solve the approximate MIPS problem efficiently.

The quantization-based techniques have shown strong performance. Particularly, in (Guo et al. 2020), the authors proposed a new family of anisotropic loss functions for the MIPS task called score-aware quantization loss, where is armed with a weight function to weigh more heavily on items with larger inner product scores a specific weight function. Therefore, the anisotropic loss is considered to be more suitable for the MIPS task so it is applied for learning product quantization. The product quantization with the anisotropic loss has achieved state-of-the-art MIPS performance. However, it has a lot of room for improvements since additive quantization has a more strong representation capacity and has been empirically shown to remarkably outperform product quantization in approximation and search accuracy. However, the codebooks in additive quantization are more difficult to learn, particularly with the anisotropic loss due to parallel-orthogonal decomposition of residual error. Therefore, it is worth investigating how to adapt anisotropic loss to additive quantization.

To this end, we propose Anisotropic Additive Quantization to combine anisotropic loss and additive quantization. To efficiently learn the codebooks, we develop a new alternating optimization algorithm after analyzing the parallel and orthogonal errors in detail. Finally, we conduct extensive 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 while guaranteeing similar retrieval efficiency.

To summarize, we deliver the following contributions:

- To the best of our knowledge, we adapt the anisotropic loss for additive quantization for the first time, to promote the advancement of approximate MIPS.
- We develop a new alternating optimization algorithm, such that the codebooks can be effectively updated and the codes of the data can be fast computed.
- The proposed Anisotropic Additive Quantization algorithm is comprehensively evaluated on three real-world datasets, where the results demonstrate that Anisotropic Additive Quantization outperforms the state-of-the-art baselines.

The rest of this paper is organized as follows: In Sec.2, the related works are discussed. In Sec.3, we introduce score-aware quantization loss as preliminary. The proposed Anisotropic Additive Quantization is in Sec.4. In Sec.5, we report our experiment results. Finally, we conclude this paper in Sec.6.

2 Related works

The MIPS problem has been studied for more than a decade. There is a large body of similarity search literature on max inner product and nearest neighbor search about hashing (Chen et al. 2019; Neyshabur and Srebro 2015; Shrivastava and Li 2014), graph search (Liu et al. 2020; Morozov and Babenko 2018; Kleinberg 2000) and quantization (Guo et al. 2020; Babenko and Lempitsky 2014), etc. Interested readers could refer to the survey (Wang et al. 2015; Li et al. 2019).

There are two main tasks 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.

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 2018; Harwood and Drummond 2016). These methods work by navigating a graph by greedily selecting the neighbor with the highest dot product.

The other task is improving the rate at which items are scored. This is typically done with quantization, which is most relevant to our work. We will pay more attention to the work related to quantization.

Quantization is widely used in state-of-the-art MIPS systems with large-scale settings. Using the quantization method, we can accomplish a more efficient calculation of the inner product by looking up the table. The time complexity required about a d-dimensional query vector with n quantization points is $\mathcal{O}(dK + nM)$, where K is the size of each quantization codebook and M is the number of codebooks. This is much faster than the $\mathcal{O}(nd)$ complexity required for exact computation. Besides, quantized datapoints take up less space in memory or on disk. This further improves the utilization of space.

Quantization-based methods usually derive multiple codebooks by minimizing the loss function between data points and the composition of codewords. It is possible to composite these codewords by concatenation and addition, such that an exponentially large codebook can be generated. 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 learned multiple codebooks in an iterative way. Besides, there are some other quantization methods like random projections (Charikar 2002; Vempala 2005; Li and Li 2019), binary quantization (He, Wen, and Sun 2013; Erin Liong et al. 2015; Dai et al. 2017).

A large body of work (Guo et al. 2020; Martinez et al. 2018; Guo et al. 2016; May et al. 2019) is dedicated to improving quantization technology to solve MIPS. Our work is a further improvement of additive quantization (Babenko and Lempitsky 2014) to make it more suitable for maximum inner product search by combining the anisotropic loss proposed in (Guo et al. 2020).

3 Preliminary

In this section, we introduce an important loss function called score-aware quantization loss which was firstly proposed in the ScaNN (Guo et al. 2020) and achieved superior performance in the maximum inner product search compared to methods using reconstruction error.

3.1 Score-Aware Quantization Loss

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

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

It takes expectation over all possible combinations of datapoints $\boldsymbol{x}$ and queries $\boldsymbol{q}$, where $\tilde{\boldsymbol{x}}$ is an approximation of $\boldsymbol{x}$ by quantization. A natural idea is that not all pairs of $(\boldsymbol{x}, \boldsymbol{q})$ 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.

Based on the above observation, the formal loss function is defined as the following:

Definition 1. *Given a datapoint $\boldsymbol{x_i}$, its quantization $\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}$, and a weight function $w : \mathbb{R} \to \mathbb{R}^+$ of the inner product score, the score-aware quantization loss with respect to a query distribution Q is defined as*

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

By defining the parallel residual error, it is the component of the residual error parallel to the datapoint x_i, which can be computed as

$$\boldsymbol{r}_{\parallel}(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) = \frac{\langle (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i), \boldsymbol{x}_i\rangle \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|^2},$$

and orthogonal residual error which can be computed as

$$\boldsymbol{r}_{\perp}(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) = (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i) - \boldsymbol{r}_{\parallel}(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i),$$

under natural statistical assumptions about the query, in (Guo et al. 2020), the authors have shown that the score-aware quantization loss can be efficiently calculated by the following theorem:

Theorem 1. *Suppose we are given a datapoint $\boldsymbol{x_i}$, its quantization $\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}$, and a weight function w. Assuming that query $\boldsymbol{q}$ is uniformly distributed in the d-dimensional unit sphere, the score-aware quantization loss equals*

$$\begin{aligned} l(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}}, w) &= h_{\parallel}(w, \|\boldsymbol{x_i}\|)\|\boldsymbol{r}_{\parallel}(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}})\|^2 \\ &+ h_{\perp}(w, \|\boldsymbol{x_i}\|)\|\boldsymbol{r}_{\perp}(\boldsymbol{x_i}, \widetilde{\boldsymbol{x}}_{\boldsymbol{i}})\|^2 \end{aligned}$$

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

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

$$h_{\perp} := \int_0^{\pi} w\left(\|\boldsymbol{x_i}\|\cos\theta\right)\sin^d\theta d\theta.$$

A simple and effective weighting function is $w(t) = \boldsymbol{I}(t \geq T)$. This weight function only considers quantization loss when the dot product is above a threshold T. In (Guo et al. 2020), the authors have shown that with this weighting function, the loss function will lead to an anisotropic weighting that more greatly penalizes error parallel with the datapoint than error orthogonal to the datapoint. Concretely,

$$h_{\parallel}(w, \|\boldsymbol{x_i}\|) \geq h_{\perp}(w, \|\boldsymbol{x_i}\|). \tag{3}$$

Thus, the score-aware quantization loss with the weighting function $w(t) = \boldsymbol{I}(t \geq T)$ is called anisotropic loss. We also use this weighting function for our method.

4 Anisotropic Additive Quantization

The additive quantization assumes to use M codebooks, $\boldsymbol{C}^{(1)}, \cdots, \boldsymbol{C}^{(M)}$, to approximate the data vector $\boldsymbol{x}$ with $\tilde{\boldsymbol{x}}$, which is defined as follows:

$$\tilde{\boldsymbol{x}} = \sum_{m=1}^{M} \boldsymbol{C}^{(m)}_{i_m(\boldsymbol{x})}.$$

where $\boldsymbol{C}^{(m)} = [\boldsymbol{C}^{(m)}_1, \ldots, \boldsymbol{C}^{(m)}_K] \in \mathbb{R}^{d\times K}$, $i_m(\boldsymbol{x})$ returns the index in the m-th codebook of the data $\boldsymbol{x}$.

Next, using anisotropic loss function $\ell(\boldsymbol{x_i}, \tilde{\boldsymbol{x}}_{\boldsymbol{i}}) = h_{i,\parallel}\left\|\boldsymbol{r}_{\parallel}(\boldsymbol{x_i}, \tilde{\boldsymbol{x}}_{\boldsymbol{i}})\right\|^2 + h_{i,\perp}\|\boldsymbol{r}_{\perp}(\boldsymbol{x_i}, \tilde{\boldsymbol{x}}_{\boldsymbol{i}})\|^2$ to obtain a new objective function for additive quantization we call the anisotropic additive 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 K d-dimensional codewords, the anisotropic additive quantization problem is to find the M codebooks that minimizes*

$$\begin{aligned} \min_{C^{(1)},\ldots,C^{(M)}} \sum_{i=1}^{n} \min_{\tilde{\boldsymbol{x}}_i \in \sum_{m=1}^{M} C^{(m)}_{i_m(x_i)}} & h_{i,\parallel}\left\|\boldsymbol{r}_{\parallel}(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)\right\|^2 \\ & + h_{i,\perp}\|\boldsymbol{r}_{\perp}(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)\|^2. \end{aligned} \tag{4}$$

4.1 Optimization Procedure

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 vector $\boldsymbol{b}_i = [\boldsymbol{b}_i^{(1)}, \cdots, \boldsymbol{b}_i^{(M)}]^\top \in \{0,1\}^{MK\times 1}$, where the k-th element $b^{(m)}_{i,k}$ of the vector $\boldsymbol{b}_i^{(m)}$ equals to 1 if $i_m(\boldsymbol{x}_i) = k$ and 0 otherwise.

If we further concatenate the codebook matrix $\boldsymbol{C}^{(m)}$ by column to obtain the matrix $\boldsymbol{C} = [\boldsymbol{C}^{(1)}, \cdots, \boldsymbol{C}^{(M)}] \in \mathbb{R}^{d\times MK}$, we can explicitly formulate the objective loss eq (4) as a function of our parameter vector $\Theta = (\boldsymbol{C}, \{\boldsymbol{b_i}\}_{[n]})$. Concretely, we have the following proposition:

Proposition 1. *The objective loss function eq* (4) *is equivalent to*

$$Loss(\boldsymbol{C}, \boldsymbol{b}) = \sum_{i=1}^{n} L^{(i)}(\boldsymbol{C}, \boldsymbol{b_i}), \tag{5}$$

with $L^{(i)}(\boldsymbol{C}, \boldsymbol{b_i})$ defined as follows:

$$\begin{aligned} L^{(i)}(\boldsymbol{C}, \boldsymbol{b_i}) = \boldsymbol{b_i}^\top \boldsymbol{C}^\top \left(\frac{h_{i,\parallel} - h_{i,\perp}}{\|\boldsymbol{x_i}\|^2}\boldsymbol{x_i}\boldsymbol{x_i}^\top + h_{i,\perp}\boldsymbol{I}\right)\boldsymbol{C}\boldsymbol{b_i} \\ - 2h_{i,\parallel}\boldsymbol{x_i}^\top \boldsymbol{C}\boldsymbol{b_i} + h_{i,\parallel}\|\boldsymbol{x_i}\|^2. \end{aligned} \tag{6}$$

Proof. Consider a single point $\boldsymbol{x}_i$ with $\boldsymbol{r}_\parallel := \boldsymbol{r}_\parallel(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) = \frac{1}{\|x\|^2}\boldsymbol{x}_i\boldsymbol{x}_i^\top(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)$ and $\boldsymbol{r}_\perp := \boldsymbol{r}_\perp(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) = \boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i - \boldsymbol{r}_\parallel$. We have that

$$\begin{aligned} \|\boldsymbol{r}_\perp\|^2 &= \left(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i - \boldsymbol{r}_\parallel\right)^\top\left(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i - \boldsymbol{r}_\parallel\right) \\ &= \|\boldsymbol{x}_i\|^2 + \|\tilde{\boldsymbol{x}}_i\|^2 - 2\boldsymbol{x}_i^\top\tilde{\boldsymbol{x}}_i - 2\boldsymbol{r}_\parallel^\top(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i) + \left\|\boldsymbol{r}_\parallel\right\|^2 \\ &= \|\boldsymbol{x}_i\|^2 + \|\tilde{\boldsymbol{x}}_i\|^2 - 2\boldsymbol{x}_i^\top\tilde{\boldsymbol{x}}_i - \left\|\boldsymbol{r}_\parallel\right\|^2, \end{aligned} \tag{7}$$

where we use the fact that $\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i = \boldsymbol{r}_\parallel + \boldsymbol{r}_\perp$ and $\boldsymbol{r}_\parallel$ is orthogonal to $\boldsymbol{r}_\perp$. We also have

$$\begin{aligned} \left\|\boldsymbol{r}_\parallel\right\|^2 &= \frac{1}{\|\boldsymbol{x}_i\|^4}\left(\boldsymbol{x}_i(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^\top\boldsymbol{x}_i\right)^\top\left(\boldsymbol{x}_i(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^\top\boldsymbol{x}_i\right) \\ &= \frac{1}{\|\boldsymbol{x}_i\|^4}\boldsymbol{x}_i^\top(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\boldsymbol{x}_i^\top\boldsymbol{x}_i(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^\top\boldsymbol{x}_i \\ &= \frac{1}{\|\boldsymbol{x}_i\|^2}\boldsymbol{x}_i^\top(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^\top\boldsymbol{x}_i \\ &= \|\boldsymbol{x}_i\|^2 + \frac{\tilde{\boldsymbol{x}}_i^\top\boldsymbol{x}_i\boldsymbol{x}_i^\top\tilde{\boldsymbol{x}}_i}{\|\boldsymbol{x}_i\|^2} - 2\boldsymbol{x}_i^\top\tilde{\boldsymbol{x}}_i. \end{aligned} \tag{8}$$

Combining equations (7) and (8), we have that

$$
\begin{aligned}
L^{(i)}(\boldsymbol{C},\boldsymbol{b_i}) :&= h_{i,\|}\left\|\boldsymbol{r}_{\|}\right\|^2 + h_{i,\perp}\left\|\boldsymbol{r}_{\perp}\right\|^2 \\
&= \tilde{\boldsymbol{x}}_i^\top\left(\left(h_{i,\|}-h_{i,\perp}\right)\frac{\boldsymbol{x}_i\boldsymbol{x}_i^\top}{\|\boldsymbol{x}_i\|^2}+h_{i,\perp}\boldsymbol{I}\right)\tilde{\boldsymbol{x}}_i \\
&\quad -2h_{i,\|}\boldsymbol{x}_i^\top\tilde{\boldsymbol{x}}_i + h_{i,\|}\left\|\boldsymbol{x}_i\right\|^2 .
\end{aligned} \tag{9}
$$

Combining above equation (9) and $\tilde{\boldsymbol{x}}_i = \boldsymbol{C}\boldsymbol{b}_i$, we get the desired result.

□

Next, we develop an iterative algorithm to optimize the anisotropic additive quantization problem. Similar to the additive quantization algorithm (Babenko and Lempitsky 2014), our algorithm iterate between minimization over the assignment $\boldsymbol{b}$ step and codebook $\boldsymbol{C}$ update step.

Update$\{\boldsymbol{b}_i\}$. It can be easily seen that $\boldsymbol{b}_i$, the index vector 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 assignment $\boldsymbol{b}_i$, we need to optimize

$$
\boldsymbol{b}_i^* = \underset{\boldsymbol{b}_i}{\arg\min}\, L^{(i)}(\boldsymbol{C},\boldsymbol{b_i}). \tag{10}
$$

This involves a combinatorial optimization problem, which suffers from high computational costs. Similar to (Zhang, Du, and Wang 2014; Martinez et al. 2016), we use the alternative optimization technique to optimize it, and solve the subvectors $\{\boldsymbol{b}_i^{(m)}\}_{m=1}^M$ alternatively. 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 equation (5).

$$
\boldsymbol{C}^* = \underset{\boldsymbol{C}}{\arg\min}\, Loss(\boldsymbol{C},\boldsymbol{b}).
$$

This is a convex quadratic minimization problem. By setting

$$
\boldsymbol{T} = \sum_i \boldsymbol{b}_i\otimes\boldsymbol{I}_d\left(h_{i,\perp}\boldsymbol{I}_d + \frac{h_{i,\|}-h_{i,\perp}}{\|\boldsymbol{x}_i\|^2}\boldsymbol{x}_i\boldsymbol{x}_i^\top\right)\boldsymbol{b}_i^\top\otimes\boldsymbol{I}_d
$$

$$
\boldsymbol{R} = \sum_i h_{i,\|}\left(\boldsymbol{b}_i\otimes\boldsymbol{I}_d\right)\boldsymbol{x}_i,
$$

where $\otimes$ denotes Kronecker product, $\boldsymbol{I}_d$ is the d-dimensional identity matrix, the closed-form solution can be given as follows:

$$
\mathrm{Vec}(\boldsymbol{C}) = \boldsymbol{T}^{-1}\boldsymbol{R},
$$

where $Vec(\boldsymbol{C})$ denotes the vectorization of the matrix $\boldsymbol{C}$, formed by stacking the columns of $\boldsymbol{C}$ into a single column vector.

It is easy to observe that in the update $\boldsymbol{C}$ step, the storage of matrix $\boldsymbol{T}$ requires $\mathcal{O}(M^2K^2d^2)$ memory. It is difficult to accept in high bits. However, higher bits tend to bring a better quantization effect, so we propose another method of updating the codebooks by using coordinate descent to alternatively update each codeword in all codebooks. Specifically, we have the following theorem:

Theorem 2. *Suppose all the index vectors $\boldsymbol{b}$ are fixed. By setting*

$$
\boldsymbol{T}_k = \sum_{i:\boldsymbol{b}_{i,k}=1}\boldsymbol{H}_i, \quad \boldsymbol{R}_k = \sum_{i:\boldsymbol{b}_{i,k}=1}\left(h_{i,\|}\boldsymbol{x}_i - \boldsymbol{H}_i\boldsymbol{S}_{i,k}\right),
$$

with $\boldsymbol{H}_i$ and $\boldsymbol{S}_{i,k}$ defined as follows:

$$
\begin{aligned}
\boldsymbol{H}_i &= \frac{h_{i,\|}-h_{i,\perp}}{\|\boldsymbol{x}_i\|^2}\boldsymbol{x}_i\boldsymbol{x}_i^\top + h_{i,\perp}\boldsymbol{I} \\
\boldsymbol{S}_{i,k} &= \sum_{j\neq k}\boldsymbol{b}_{i,j}\boldsymbol{C}_j,
\end{aligned} \tag{11}
$$

the update formula for codeword $\boldsymbol{C}_k$ can be given as follows:

$$
\boldsymbol{C}_k = \boldsymbol{T}_k^{-1}\boldsymbol{R}_k, \forall k\in\{1,2,\ldots,M*K\}. \tag{12}
$$

When we are updating $\boldsymbol{C}_k$, we consider the other $C_i, i\neq k$ as constants.

Proof. Ignoring the constant term $h_{i,\|}\|\boldsymbol{x_i}\|^2$ in equation (6), we have

$$
\begin{aligned}
L_i &= \boldsymbol{b}_i^\top\boldsymbol{C}^\top\boldsymbol{H}_i\boldsymbol{C}\boldsymbol{b}_i - 2h_{i,\|}\boldsymbol{x}_i^\top\boldsymbol{C}\boldsymbol{b}_i \\
&= \left(\sum_{k=1}^{MK}\boldsymbol{b}_{i,k}\boldsymbol{C}_k^\top\right)\boldsymbol{H}_i\left(\sum_{k=1}^{MK}\boldsymbol{b}_{i,k}\boldsymbol{C}_k\right) - 2h_{i,\|}\boldsymbol{x}_i^\top\sum_{k=1}^{MK}\boldsymbol{b}_{i,k}\boldsymbol{C}_k.
\end{aligned}
$$

Then,

$$
\nabla_{\boldsymbol{C}_k}L_i = 2\boldsymbol{b}_{i,k}^2\boldsymbol{H}_i\boldsymbol{C}_k + 2\boldsymbol{b}_{i,k}\boldsymbol{H}_i\sum_{j\neq k}\boldsymbol{b}_{i,j}\boldsymbol{C}_j - 2h_{i,\|}\boldsymbol{b}_{i,k}\boldsymbol{x}_i,
$$

Note that here $\boldsymbol{b}_{i,k}$ is only equal to 1 or 0 for all k $\in[M*K]$, $\boldsymbol{H}_i$ is defined in equation (11).

Since the equation (3) showed that $h_{\|}\geq h_{\perp}$, the loss function is a convex quadratic function. Thus, let $\nabla_{\boldsymbol{C}_k}Loss = 0$, we have

$$
\begin{aligned}
\boldsymbol{C}_k &= \left(\sum_{i:\boldsymbol{b}_{i,k}=1}\boldsymbol{H}_i\right)^{-1}\left[\sum_{i:\boldsymbol{b}_{i,k}=1}\left(h_{i,\|}\boldsymbol{x}_i - \boldsymbol{H}_iS_{i,k}\right)\right] \\
&= T_k^{-1}R_k,
\end{aligned}
$$

where we use the fact that $\nabla_{\boldsymbol{C}_k}Loss = \sum_{i=1}^n\nabla_{\boldsymbol{C}_k}L_i$, the $Loss$ is defined in equation (5) and $S_{i,k}$ is defined in equation (11).

□

Initialization. As the process of updating each codeword requires other codewords, the initiation of the learning process needs initial codebooks, unable to only set the assignment variables randomly like AQ (Babenko and Lempitsky 2014). In most of the experiments below, we initialize the learning process by selecting K random datapoints or performing initialization by codebooks obtained within anisotropic production quantization. (a PQ codeword can be turned into a full-length AQ vector, by padding it with zero chunks)

The complete optimization procedure is the following:

1. (Initialization Step) Select K random datapoints or perform initialization by codebooks obtained within anisotropic production quantization.
2. (Partition Assignment Step) For each datapoint $\boldsymbol{x}_i$, update $\boldsymbol{b}_i$ by using the value of current codebooks that minimizes the anisotropic loss eq (10).
3. (Codebook Update Step) Optimize the loss function by alternatively update each codeword eq (12) in all codebooks while keeping every codebooks partitions constant.
4. Repeat Step 2 and Step 3 until convergence to a fixed point or the maximum number of iteration is reached.

4.2 Complexity Analysis

The encoding time complexity of each data point is $\mathcal{O}(I_e MKd)$, where I_e is the maximum number of iteration rounds in updating $\boldsymbol{b}_i$. In our experiments, I_e is set to 3. The mean time complexity for updating each codeword requires $\mathcal{O}(nd^2/K + d^3)$, The latter part is due to matrix inverse $\boldsymbol{T}_k^{-1}$, the former part is other matrix computation related to $\boldsymbol{H}_i$ and $\boldsymbol{S}_{i,k}$. The complete update of codebooks requires $\mathcal{O}(I_c nMd^2 + I_c MKd^3)$ time complexity, where I_c is the maximum number of iteration rounds in updating $\boldsymbol{C}$. The memory we need during all update processes is $\mathcal{O}(d^2 + MKd)$, which is nothing to reckon with.

5 Experiments

In this section, we conduct experiments on three real-world datasets to show our method leads to improved performance on maximum inner product search. Firstly, we will introduce the three datasets in our experiments. After comparing loss and relative error with ScaNN, we further combine the inverted indexing structure to investigate the performance and efficiency of the proposed method. Finally, we conducted experiments to compare with baselines.

All the quantization methods are implemented in Python and all experiments are conducted in a Linux server with 3.00GHZ intel GPU and 300G main memory.

5.1 Datasets

We use three real-world datasets to evaluate our algorithm. As maximum inner product search is often used in recommender systems, two datasets we use, LastFM and EchoNest are famous music recommendation datasets. In addition, we also compare our algorithms on Glove1.2M, which is used in ScaNN (Guo et al. 2020) for evaluation.

LastFM dataset collected $357,847$ items, $156,122$ users scored on it. EchoNest dataset collected 260417 items, 766882 users scored on it. We use matrix decomposition to train them into 32-dimensional embedding vectors as described in (Lian et al. 2015), and $10,000$ users are randomly selected as queries. The Glove1.2M is a collection of 1.2 million 100-dimensional word embeddings trained as described in (Pennington, Socher, and Manning 2014). The Glove dataset is meant to be used with a cosine distance similarity metric. MIPS is equivalent to cosine similarity search when all datapoints are equal-norm. Following (Guo et al. 2020), we adopt our technique to cosine similarity search by unitnormalizing all datapoints on Glove1.2M.

5.2 Comparison with ScaNN

Loss and relative error We start by directly comparing the score-aware loss between the proposed method and the ScaNN on the same anisotropic quantization threshold in LsatFM.

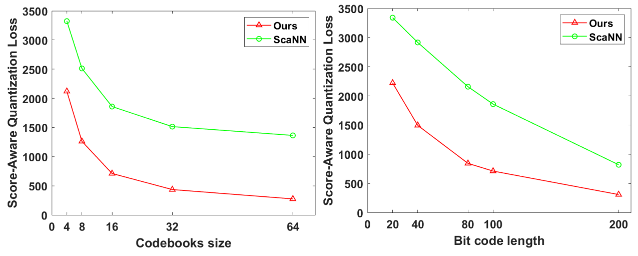

Figure 1: Score-aware loss on LsatFM for the proposed method and ScaNN with different code bit lengths and codebooks sizes.

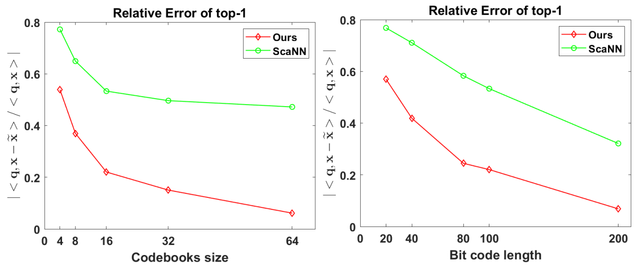

Figure 2: The relative error of inner product estimation for true Top-1 on LsatFM across multiple code bit lengths and codebooks size settings.

As the number of codebooks in product quantization must divide the dimensionality of the data, in order to obtain as many experimental results as possible, in this subsection we apply matrix decomposition on LsatFM to obtain 100-dimensional embedding vectors.

In Figure 1, we plot score-aware loss as a function of the code length or codebooks size for the LsatFM. It can be seen that for all code lengths and codebooks size, our method score-aware loss is considerably lower than for the ScaNN.

Next we look at the accuracy of the estimated top-1 inner product as measured by relative error $|\frac{\langle \boldsymbol{q}, \boldsymbol{x} - \widetilde{\boldsymbol{x}} \rangle}{\langle \boldsymbol{q}, \boldsymbol{x} \rangle}|$. This is important in application scenarios where an accurate estimate of $\langle \boldsymbol{q}, \boldsymbol{x} \rangle$ is needed, such as softmax approximation, where the inner product values are often logits later used to compute probabilities.

We see in Figure 2 that our method leads to smaller relative error overall bit settings and codebooks size. Given these encouraging results, we further investigate the performance and efficiency of the proposed method.

Combined with Inverted Indexing Structure As the time complexity $\mathcal{O}(nM)$ of table lookup is related to n

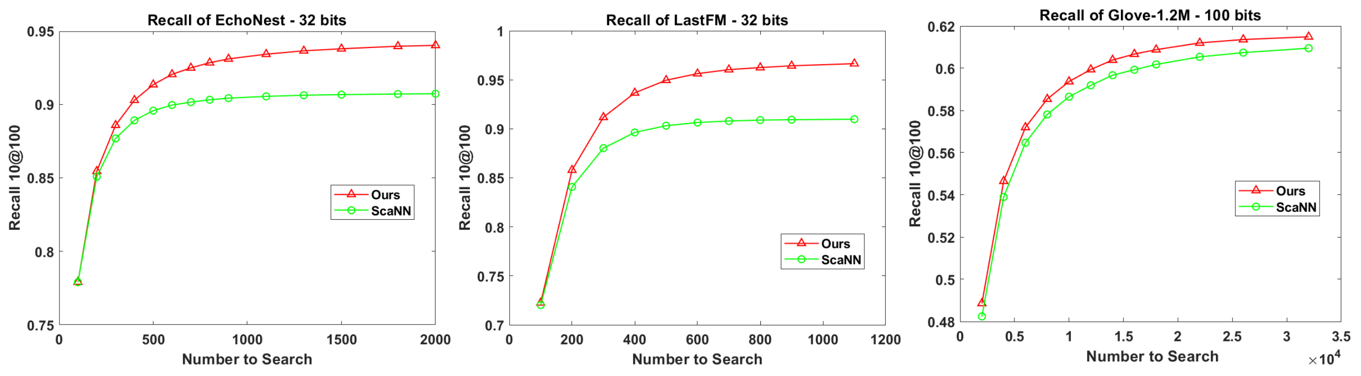

Figure 3: Recall curves when returning 100 neighbors (Recall10@100) with varying the number of searched items.

when the number of data is large, it will suffer a large time consumption, in order to improve the efficiency, we further combine the inverted indexing structure (Jegou, Douze, and Schmid 2010).

It clusters the data first by the coarse quantizer so that the query only needs to find some of the most relevant centroids. It only takes a small amount of time $\mathcal{O}(K_v d)$ to find the relevant centroids, which reduces the time complexity to $\mathcal{O}(n_q M)$ when calculating the inner product by using lookup tables, where K_v is the number of centroids in the coarse quantizer, n_q is the number of items to search about the specified query. Usually n_q is much less than n. Thus, combined with inverted indexing, the algorithm is more suitable for large-scale searches and can return more results faster.

Under the inverted structure, the number of search items is the main factor affecting the performance of the algorithm. We investigate its effect in this subsection while further comparing it with ScaNN.

In this experiment, we set ground-truth to 10, and the evaluation criteria used is recall10@100 which indicates the portion of the ground-truth being included in the top-100 results. Note that the 100 neighbors here are only selected according to the ADC score (Jegou, Douze, and Schmid 2010), if we need a higher recall, we can return more neighbors according to the ADC score, then rerank the results by the exact inner product.

In Figure 3, we can see that the proposed method outperforms ScaNN in all different search scales. Besides, we also see that how to reasonably determine the number of items to search, which will be used in the Comparison with Baselines.

Speed Benchmarks The efficiency of the algorithm is crucial in the MIPS. In this subsection, we further investigate the efficiency of the algorithm. For a fair comparison, we use the same inverted indexing structure, and the same ADC computation based on the inner product for ScaNN (Guo et al. 2020).

In time complexity, the AQ algorithm takes more time to build the lookup table than the PQ algorithm, but this is not critical because their other query processes, such as finding the coarse quantizer cluster centroids to search for, calculate the inner product by using lookup tables, and sorting ADC scores, are the same. Our experiments also confirm it.

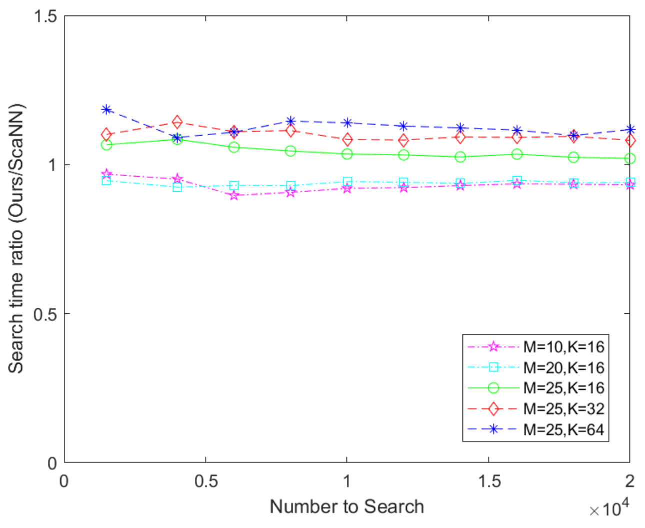

Figure 4: Search time ratio with varying search sizes for different codes length M, codebooks size K on the Glove dataset.

From the Figure 4, it can be seen that most of the ratios are concentrated around 1 even at very high bits. The reason why the ratio appears less than 1 is that the PQ algorithm builds lookup tables on multiple subspaces, while the AQ algorithm builds lookup tables only on a full d-dimensional space, which has less expenditure other than computation. It indicates that the two methods have similar search times, while according to the Figure 3, the proposed method has a better performance in terms of recall.

5.3 Comparison with Baselines

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

- **ScaNN (Guo et al. 2020)**, is the state-of-the-art MIPS quantization method. ScaNN and our method are compared on the same number of codebooks, same number

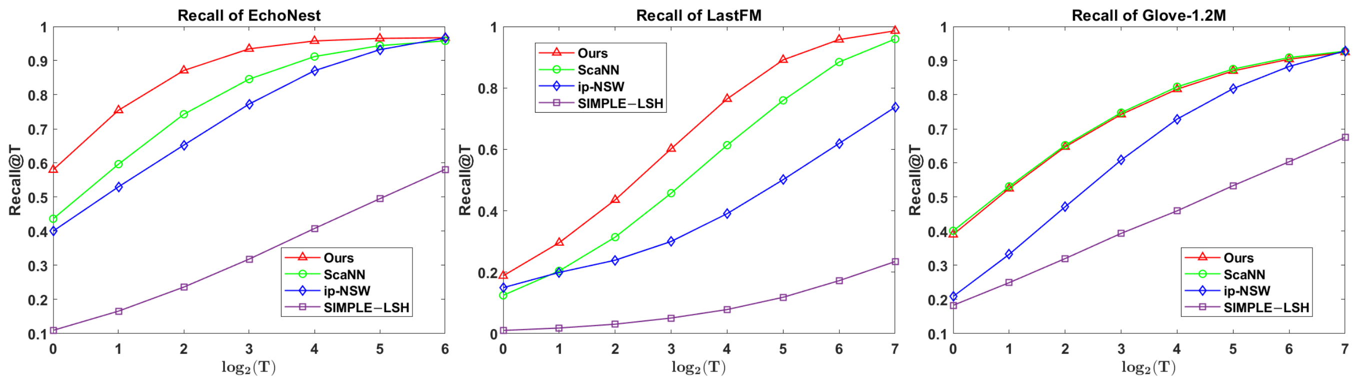

Figure 5: Recall 1@T curve comparing with baselines on MIPS tasks.

of codewords, and same anisotropic quantization threshold.

- **SIMPLE-LSH (Neyshabur and Srebro 2015)**, is the representative work of the MIPS hashing method, which transforms the input vectors such that the MIPS problem becomes equivalent to the L_2NNS problem in the transformed space. 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).
- **ip-NSW (Morozov and Babenko 2018)**, is the representative work of the MIPS graph method, which solves the MIPS problem with the usage of similarity graphs. We used the author's implementation and the recommended parameters. In addition, for a reasonable comparison, when constructing the graph, we set the number of edges per vertex as the same as the number of codebooks in quantization methods, the size of the dynamic candidate list is set to 1024.

Settings In addition to the above descriptions, for the two music dataset, we use 8 codebooks in quantization methods, with anisotropic quantization threshold T set to 0.1 times the mean norm of the datapoints on EchoNest and 0.05 times the mean norm of the datapoints on LastFM. The Glove dataset is consistent with ScaNN, using 50 codebooks with T set to 0.2. The number of codewords is set to 16 except for EchoNest, which is set to 128.

Results In terms of efficiency, due to the different implementation languages (e.g. C and C++ are more efficient than Python.), the programmers' skills (e.g. whether programmers use parallel computation mode), the characteristics of different methods (e.g. SIMPLE-LSH needs to search all datapoints, other methods only need to search partial datapoints), a fair comparison of efficiency between different methods is difficult to achieve. We compare only with the ScaNN which is also a quantization method and is efficient. In section 5.2, we have conducted a comprehensive efficiency comparison with ScaNN. The results are presented in Figure 4.

In terms of accuracy, we report the experiment by the recall@T measure, defined as a probability (computed over a number of queries) that the set of T closest neighbors returned by algorithms contains the true nearest neighbor. We generate ground-truth results using brute force search and compare the neighbors returned by each method against ground-truth. The results are shown in the form of recall@T-vs-$log_2(T)$ curves.

From the recall curves Figure 5, it is clear that our method outperforms existing graph and hashing methods. Our method achieves similar results to ScaNN on the glove dataset, but significantly outperforms ScaNN on the other datasets. It shows that our algorithm returns more accurate neighbors.

6 Conclusions

In this paper, we propose a quantization method called Anisotropic Additive Quantization to handle fast inner product search. We develop a new alternating optimization algorithm, such that the codebooks can be effectively updated and the codes of the data can be fast computed. Finally, we conducted extensive experiments on three real-world datasets, investigating the relative error, recall, and efficiency of the proposed method. The results show that the proposed method outperforms the state-of-the-art baselines with respect to approximate search accuracy while guaranteeing similar retrieval efficiency.

Acknowledgments

The work was supported by grants from the National Natural Science Foundation of China (No. 62022077 and 61976198) and the Fundamental Research Funds for the Central Universities (No. WK2150110017).

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.

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

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.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, 39–46.

Dai, B.; Guo, R.; Kumar, S.; He, N.; and Song, L. 2017. Stochastic generative hashing. In *International Conference on Machine Learning*, 913–922. PMLR.

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

Erin Liong, V.; Lu, J.; Wang, G.; Moulin, P.; and Zhou, J. 2015. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2475–2483.

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.

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.

He, K.; Wen, F.; and Sun, J. 2013. 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*, 2938–2945.

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

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.

Kleinberg, J. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 163–170.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.

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*.

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.

Li, W.; Zhang, Y.; Sun, Y.; Wang, W.; Li, M.; Zhang, W.; and Lin, X. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8): 1475–1488.

Li, X.; and Li, P. 2019. Random Projections with Asymmetric Quantization. In *NeurIPS*, 10857–10866.

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.

Liu, J.; Yan, X.; Dai, X.; Li, Z.; Cheng, J.; and Yang, M.-C. 2020. Understanding and improving proximity graph based maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 139–146.

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.

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.

Martinez, J.; Clement, J.; Hoos, H. H.; and Little, J. J. 2016. Revisiting additive quantization. In *European Conference on Computer Vision*, 137–153. Springer.

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

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.

Mussmann, S.; and Ermon, S. 2016. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*, 2587–2596. PMLR.

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.

Pritzel, A.; Uria, B.; Srinivasan, S.; Badia, A. P.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural episodic control. In *International Conference on Machine Learning*, 2827–2836. PMLR.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

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

Wang, J.; Liu, W.; Kumar, S.; and Chang, S.-F. 2015. Learning to hash for indexing big data—A survey. *Proceedings of the IEEE*, 104(1): 34–57.

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

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.

Yen, I. E.-H.; Kale, S.; Yu, F.; Holtmann-Rice, D.; Kumar, S.; and Ravikumar, P. 2018. Loss decomposition for fast learning in large output spaces. In *International Conference on Machine Learning*, 5640–5649. PMLR.

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.