Norm-Explicit Quantization: Improving Vector Quantization for Maximum Inner Product Search

Xinyan Dai,* Xiao Yan,* Kelvin K. W. Ng, Jie Liu, James Cheng
The Chinese University of Hong Kong
{xydai, xyan, kwng6, jliu, jcheng}@cse.cuhk.edu.hk

Abstract

Vector quantization (VQ) techniques are widely used in similarity search for data compression, computation acceleration and etc. Originally designed for Euclidean distance, existing VQ techniques (e.g., PQ, AQ) explicitly or implicitly minimize the quantization error. In this paper, we present a new angle to analyze the quantization error, which decomposes the quantization error into norm error and direction error. We show that quantization errors in norm have much higher influence on inner products than quantization errors in direction, and small quantization error does not necessarily lead to good performance in maximum inner product search (MIPS). Based on this observation, we propose norm-explicit quantization (NEQ) — a general paradigm that improves existing VQ techniques for MIPS. NEQ quantizes the norms of items in a dataset explicitly to reduce errors in norm, which is crucial for MIPS. For the direction vectors, NEQ can simply reuse an existing VQ technique to quantize them without modification. We conducted extensive experiments on a variety of datasets and parameter configurations. The experimental results show that NEQ improves the performance of various VQ techniques for MIPS, including PQ, OPQ, RQ and AQ.

1 Introduction

Given a dataset $\mathcal{X} \subset \mathbb{R}^d$ that contains n vectors (also called items) and a query $q \in \mathbb{R}^d$, maximum inner product search (MIPS) finds an item x^* that has the largest inner product with the query,

$$x^* = \arg\max_{x \in \mathcal{X}} q^\top x. \quad (1)$$

The definition of MIPS can be easily extended to top-k inner product search, which is used more commonly in practice. MIPS has many important applications such as recommendation based on user and item embeddings (Koren, Bell, and Volinsky 2009), multi-class classification with linear classifier (Dean et al. 2013), and object matching in computer vision (Felzenszwalb et al. 2010). Recently, MIPS is also used for Bayesian interference (Mussmann and Ermon 2016), memory network training (Chandar et al. 2016) and reinforcement learning (Jun et al. 2017).

Vector quantization (VQ). VQ quantizes items in the dataset with M codebooks $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^M$. Each codebook $\mathcal{C}^m$ contains K codewords and each codeword is a d-dimensional vector, i.e., $\mathcal{C}^m = \{c^m[1], c^m[2], ..., c^m[K]\}$, $c^m[k] \in \mathbb{R}^d$ for $1 \le m \le M$ and $1 \le k \le K$. Denote i_x^m as the index of the codeword in codebook $\mathcal{C}^m$ that item x maps to, then x is approximated by $\tilde{x} = \sum_{m=1}^{M} c^m[i_x^m]$. Therefore, the inner product between query q and item x, i.e., $q^\top x$, is approximated by $q^\top \tilde{x} = \sum_{m=1}^{M} q^\top c^m[i_x^m]$. There are a number of VQ algorithms with different quantization strategies and codebook learning procedures, such as product quantization (PQ) (Jégou, Douze, and Schmid 2011), optimized product quantization (OPQ) (Ge et al. 2013), residual quantization (RQ) (Chen, Guan, and Wang 2010) and additive quantization (AQ) (Babenko and Lempitsky 2014). We describe them in greater details in Section 2.

VQ can be used for *data compression*, *fast inner product computation* and *candidate generation* in MIPS. For data compression, the M codeword indexes $\{i_x^1, i_x^2, ..., i_x^M\}$ is stored instead of the original d-dimensional vector x, which enables storing very large datasets (e.g., with 1 billion items) in the main memory of a single machine (Johnson, Douze, and Jégou 2017). When the inner products between query q and all codewords are precomputed and stored in look-up tables, the approximate inner product of an item (i.e., $q^\top \tilde{x}$) can be computed with a complexity of $O(M)$ instead of $O(d)$. With two codebooks, VQ can use the efficient multi-index algorithm (Babenko and Lempitsky 2012) to generate candidates for MIPS. Note that VQ is orthogonal to existing MIPS algorithms, such as tree-based methods (Koenigstein, Ram, and Shavitt 2012; Ram and Gray 2012), locality sensitive hashing (LSH) based methods (Neyshabur and Srebro 2015; Shrivastava and Li 2014), proximity graph based method (Morozov and Babenko 2018) and pruning based methods (Li et al. 2017; Teflioudi, Gemulla, and Mykytiuk 2015). These algorithms focus on generating good candidates for MIPS, while VQ focuses on data compression and computation acceleration. Actually, VQ can be used as a component of these algorithms

*Co-first authors are ranked alphabetically. Correspondence to Xiao Yan.

for compression and fast computation as in (Douze, Sablayrolles, and Jégou 2018).

When using VQ for similarity search, the primary performance indicator is the quality of the similarity value calculated with the codebook-based approximation $\tilde{x}$. Existing VQ techniques were primarily designed for Euclidean nearest neighbor search (Euclidean NNS) instead of MIPS. They minimize the quantization error ($\|x - \tilde{x}\|$) explicitly or implicitly because it provides an upper bound for the error of the codebook based approximate Euclidean distance, i.e., $|\|x - q\| - \|\tilde{x} - q\|| \leq \|x - \tilde{x}\|$. However, inner product is different from Euclidean distance in several important aspects. In particular, inner product does not satisfy the triangle inequality and non-negativity. The inner product between an item and itself (i.e., $x^\top x$) is not guaranteed to be the largest, while self-distance (i.e., $\|x - x\|$) is guaranteed to be the smallest for Euclidean distance. These differences prompt us to ask the following two questions: *Does minimizing quantization error necessarily lead to good performance for MIPS? Do we need a different design principle for VQ techniques when used for MIPS (than for Euclidean NNS)?*

To answer these questions, we start by analyzing the quantization errors of VQ techniques from a new angle. Instead of treating the quantization error $\|x - \tilde{x}\|$ as a whole, we decompose it into two parts: *norm error* ($\|x\| - \|\tilde{x}\|$) and *angular error* ($1 - \frac{x^\top \tilde{x}}{\|x\|\|\tilde{x}\|}$). We found that norm error has a more significant influence on inner product than angular error. Based on this observation, we propose *norm-explicit quantization* (*NEQ*), which quantizes the norm $\|x\|$ and the unit-norm direction vector $x/\|x\|$ separately. Quantizing norm explicitly using dedicated codebooks allows to reduce errors in norm, which is beneficial for MIPS. The direction vector can be quantized using existing VQ techniques without modification. NEQ is simple in that the complexity of both codebook learning and approximate inner product computation is not increased compared with the baseline VQ technique used for direction vector quantization. More importantly, NEQ is general and powerful in that it can significantly boost the performance of many existing VQ techniques for MIPS.

We evaluated NEQ on four popular benchmark datasets, where the cardinalities of the datasets range from 17K to 100M and their norm distributions are significantly different. The experimental results show that NEQ improves the performance of PQ (Jégou, Douze, and Schmid 2011), OPQ (Ge et al. 2013), RQ (Chen, Guan, and Wang 2010) and AQ (Babenko and Lempitsky 2014) for MIPS consistently on all datasets and parameter configurations (e.g., the number of codebooks and the required top-k items). NEQ also significantly outperforms the state-of-the-art LSH-based MIPS methods and provides better time-recall performance than the graph-based ip-NSW algorithm.

Contributions. Our contributions are three-folds. First, we challenge the common wisdom of minimizing the quantization error in existing VQ techniques and questioned whether it is a suitable design principle for MIPS. Second, we show that norm error has more significant influence on inner product than angular error, which leads to a more suitable design principle for MIPS. Third, we propose NEQ, a general framework that can be seamlessly combined with existing VQ techniques and consistently improves their performance for MIPS, which is beneficial to applications that involve MIPS.

2 Related Work

In this section, we introduce some popular VQ techniques to facilitate further discussion and discuss the relation between NEQ and some related work.

PQ and OPQ. PQ (Jégou, Douze, and Schmid 2011) first generates M sub-datasets $\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^M$ for the original dataset, each containing $d' = d/M$ features from all items. K-means is used to learn a codebook on each sub-dataset independently and each codeword is a d'-dimensional vector. An item x is approximated by the concatenation of its corresponding codewords from each of the codebooks, i.e., $\tilde{x} = [c^1[i_x^1], c^2[i_x^2], ..., c^M[i_x^M]]$. OPQ (Ge et al. 2013) uses an orthonormal matrix R to rotate the items by Rx before applying PQ. OPQ achieves lower quantization error when the features are correlated or some features have larger variance than others. However, codebook learning is more complex for OPQ as it involves multiple rounds of alternating optimization of the codebooks and the rotation matrix R.

RQ and AQ. Different from PQ and OPQ, in RQ (Chen, Guan, and Wang 2010) every codebook covers all features and each codeword is a d-dimensional vector. The original data are used to train the first codebook with K-means and the residues ($x - c^1[i_x^1]$) are used to train the second codebook. This process is recursive in that the m-th codebook is trained with the residues from the previous $(m-1)$ codebooks. Similar to RQ, each codebook in AQ (Babenko and Lempitsky 2014) also covers all features. AQ improves RQ by jointly optimizing all the M codebooks. Beam search is used for encoding (finding the optimal codeword indexes of an item in the codebooks) with given codebooks and a least-square formulation is used to optimize the codebooks under given encoding.

In addition to the VQ techniques introduced above, there are many other VQ techniques, such as CQ (Zhang, Du, and Wang 2014), TQ (Babenko and Lempitsky 2015) LOPQ (Kalantidis and Avrithis 2014) and LSQ (Martinez et al. 2016). Although these VQ techniques differ in their quantization strategies (e.g., partitioning the features or not) and the codebook learning algorithms (e.g., K-means or alternating minimization), all of them explicitly or implicitly minimize the quantization error $\|x - \tilde{x}\|$, which is believed to provide good performance for Euclidean NNS. In the next section, we show that this principle does not apply for MIPS.

Existing work. Similar to some other VQ algorithms used for similarity search (e.g., PQ and RQ), the prototype of NEQ can also be found in earlier researches on signal compression. The shape-gain algorithm (Gersho and Gray 1991) separately quantizes the magnitude and direction of a signal to achieve efficiency with some loss in accuracy. Instead of hurting accuracy, NEQ shows that the separate quantization of norm and direction actually improves performance for MIPS. A recent work, multi-scale quantization (Wu et al. 2017) also explicitly quantizes the norm and the motivation is to better reduce the quantization error when the dynamic range (i.e., spread of the norm distribution) is large. In contrast, NEQ

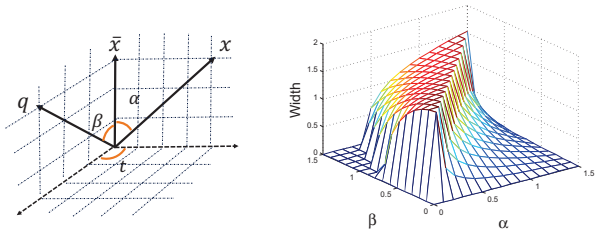

Figure 1: Illustration of Theorem 1

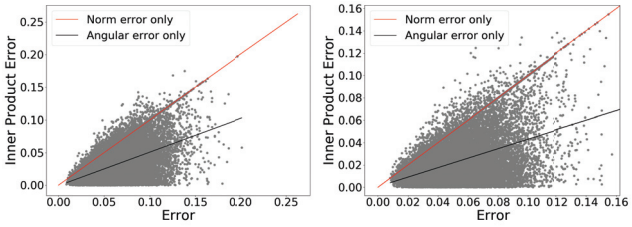

Figure 2: Influence of norm error and angular error on inner product for PQ (left) and RQ (right), all red points reside on the red line

does not try to minimize the quantization error and is not limited to the case that the data have large dynamic range. In fact, NEQ still provides significant performance improvement even if items in the dataset have almost identical norm.

3 Analysis of Quantization Error for MIPS

For Euclidean distance, quantization error provides an upper bound on the error in the approximate Euclidean distance due to the triangle inequality, i.e., $|\|x-q\| - \|\tilde{x}-q\|| \leq \|x-\tilde{x}\|$. Therefore, almost all VQ techniques try to minimize the quantization error when learning the codebooks. For approximate inner product, $\|x-\tilde{x}\|$ provides a trivial error bound because $\left|q^\top x - q^\top \tilde{x}\right| \leq \|q\|\|x-\tilde{x}\|$. As high-dimensional vectors tend to be orthogonal to each other (Cai, Fan, and Jiang 2013), the bound is loose and $q^\top(x-\tilde{x})$ can be significantly smaller than $\|q\|\|x-\tilde{x}\|$. Thus, we need to understand the influence of quantization error on inner product from a new angle. The exact inner product and its codebook-based approximation can be expressed as,

$$q^\top x = \|x\| \cdot \left(q^\top \frac{x}{\|x\|}\right) and \;\; q^\top \tilde{x} = \|\tilde{x}\| \cdot \left(q^\top \frac{\tilde{x}}{\|\tilde{x}\|}\right) \tag{2}$$

in which $x/\|x\|$ and $\tilde{x}/\|\tilde{x}\|$ are the unit-norm direction vectors of x and $\tilde{x}$, respectively. It can be observed from (2) that the accuracy of the approximate inner product depends on two factors, i.e., the quality of norm approximation ($\|\tilde{x}\|$ for $\|x\|$) and the quality of direction vector approximation ($\tilde{x}/\|\tilde{x}\|$ for $x/\|x\|$). *But how do the two factors affect the quality of approximate inner product? Does one have greater influence than the other?* To facilitate further analysis, we formally define inner product error, norm error, and angular error as follows.

Definition 1. *For an item x and its codebook-based approximation $\tilde{x}$, given a query q, the inner product error u, norm error γ, and angular error η are given as:*

$$u = \left|\frac{q^\top x - q^\top \tilde{x}}{q^\top x}\right|, \quad \gamma = \left|\frac{\|x\| - \|\tilde{x}\|}{\|x\|}\right|, \quad \eta = 1 - \frac{x^\top \tilde{x}}{\|x\|\|\tilde{x}\|}.$$

We define the inner product error and norm error as ratios over the actual values to exclude the scaling effect of q and $\|x\|$. For angular error, $\eta = 0$ if x and $\tilde{x}$ are perfectly aligned in direction.

To analyze the influence of norm error and angular error individually, we need to exclude the influence of the other. Therefore, we used the approximation $\hat{x} = \|\tilde{x}\| \cdot \frac{x}{\|x\|}$, which is accurate in direction, to calculate inner product error caused by norm approximation. Similarly, we used $\bar{x} = \|x\| \cdot \frac{\tilde{x}}{\|\tilde{x}\|}$, which is accurate in norm, to calculate the inner product error caused by direction approximation. A norm error of γ will cause an inner product error $u = \gamma$ when there is no angular error as $u = \left|\frac{q^\top x - q^\top \hat{x}}{q^\top x}\right| = \left|\frac{\|x\| - \|\tilde{x}\|}{\|x\|}\right|$. Theorem 1 formally establishes that there are cases that an angular error η results in an inner product error $u < \eta$.

Theorem 1. *For an item x, its approximation $\bar{x}$ which is accurate in norm but inaccurate in direction, and a query q, denote the angle between x and $\bar{x}$ as α and assume $\alpha \in (0, \pi/2)$, the angle between $\bar{x}$ and q as β and assume $\beta \in (0, \pi/2)$, the angle between the two planes defined by $(x, \bar{x})$ and $(\bar{x}, q)$ as t. The inner product error $\left|\frac{q^\top x - q^\top \bar{x}}{q^\top x}\right|$ is not larger than the angular error $1 - \frac{x^\top \bar{x}}{\|x\|\|\bar{x}\|}$ if angle t satisfies $\frac{\cos(\beta)}{\sin(\alpha)\sin(\beta)}\left[\frac{1}{2-\cos(\alpha)} - \cos(\alpha)\right] \leq \cos(t) \leq \frac{\cos(\beta)}{\sin(\alpha)\sin(\beta)}\left[\frac{1}{\cos(\alpha)} - \cos(\alpha)\right]$.*

We provide an illustration of the vectors in Theorem 1 in Figure 1 and the proof can found in the supplementary material. We also plot the width of the feasible region of t in the range of $(0, \pi/2)$, i.e., the difference between the maximum value and minimum value for Theorem 1 to hold, under different configurations of α and β in Figure 1. The results show that when both α and β are small and $\alpha < \beta$, for almost all $t \in (0, \pi/2)$, the the inner product error is smaller than the angular error. The required conditions are not very restrictive as we analyze below.

We consider an item x having large inner product with q as it is easy to distinguish items having large inner products with the query from those having small inner products. To achieve good performance, a VQ method should be able to distinguish items having large but similar inner products with the query. Firstly, the conditions that $\alpha \in (0, \pi/2)$ and α is small are easy to satisfy as $\bar{x}$ is the codebook based approximation of x and it should have a small angle with x. Secondly, as x has a large inner product with query q, its approximation $\bar{x}$ should also have a small angle with q, therefore the condition that $\beta \in (0, \pi/2)$ and β is small is likely to hold. Finally, as $\bar{x}$ is trained to approximate x and q is not, $\alpha < \beta$ is again easy to satisfy. As q, x and $\bar{x}$ have small angles with each other, t is likely to fall in $(0, \pi/2)$.

Theorem 1 is also supported by the following experiment

on the SIFT1M dataset [1]. We used 10,000 randomly selected queries and the errors are calculated on their ground-truth top-20 MIPS results [2] in the dataset. We experimented with PQ and RQ using 8 codebooks each containing 256 codewords. For each item-query pair (x, q), we plot two points in Figure 2. One (in red) shows the norm error and the inner product error caused by inaccurate norm (using $\hat{x}$). The other (in gray) shows the angular error and the inner product error caused by inaccurate direction vector (using $\bar{x}$). The results show that all red points reside on the line with a slope of 1, which verifies that a norm error of γ will cause an inner product error $u = \gamma$. In contrast, most of the gray points are below the red line, which means that an angular error η usually results in an inner product error $u < \eta$. We fitted a line for the gray points and the slopes for PQ and RQ are 0.510 and 0.426, respectively. The Pearson's correlation coefficients between norm error and inner product error are 1 for both PQ and RQ. While the Pearson's correlation coefficients between angular error and inner product error are 0.475 and 0.382 for PQ and RQ, respectively. We also plot the influence of norm error and angular error on Euclidean distance in the supplementary material [3], which shows that angular error has larger influence than norm error on Euclidean distance.

In conclusion, the results in this section show that norm error has more significant influence on inner product than angular error in most cases. Therefore, to improve the performance of VQ techniques for MIPS, we should reduce quantization errors in norm. To achieve this goal, we can modify the formulations of the codebook learning problem in existing VQ algorithms to consider norm error (e.g., incorporating norm error into the cost function or constraints). However, this methodology has a problem in generality as we need to modify each VQ algorithm individually. In contrast, norm-explicit quantization (NEQ) uses the fact that norm is a scalar summary of the vector and explicitly quantizes it to reduce error. As a result, NEQ can be naturally combined with any VQ algorithm by using it to quantize the direction vector.

4 Norm-Explicit Quantization

Existing VQ techniques try to minimize the quantization error and do not allow explicit control of norm error and angular error. However, MIPS could benefit from methods that explicitly reduce the error in norm because accurate norm is important for MIPS. Therefore, the core idea of NEQ is to quantize the norm $\|x\|$ and the direction vector $\frac{x}{\|x\|}$ of the items separately. The norm is encoded explicitly using separate codebooks to achieve a small error, while the direction vector can be quantized using an existing VQ quantization technique without modification. To be more specific, the M codebooks in NEQ are divided into two parts. The first M' codebooks $\mathcal{L}^1, \mathcal{L}^2, ..., \mathcal{L}^{M'}$ are norm codebooks, in which each codeword $l^m[k] \in \mathbb{R}$ for $1 \leq m \leq M'$ and $1 \leq k \leq K$. The other $M - M'$ codebooks $\mathcal{C}^{M'+1}, \mathcal{C}^{M'+2}, .., \mathcal{C}^{M}$ are vector codebooks for the direction vector. In NEQ, the codebook based approximation $\tilde{x}$ of x can be expressed as,

$$\tilde{x} = \left(\sum_{m=1}^{M'} l^m[i_x^m] \right) \cdot \left(\sum_{m=M'+1}^{M} c^m[i_x^m] \right), \qquad (3)$$

in which $i_x^1, i_x^2, ..., i_x^M$ are the codeword indexes of x in the codebooks. According to (3), NEQ-based approximate inner product $q^\top \tilde{x}$ can be calculated using Algorithm 1. Lines 4-6 reconstruct the approximate norm of x and Lines 7-9 compute the inner product between q and the approximate direction vector of x. Note that the inner product computation $q^\top c^m[i_x^m]$ in Line 8 can be replaced by table lookup when the inner products between q and the codewords are precomputed.

Algorithm 1 NEQ: Approximate Inner Product Calculation

1: **Input:** Query q, M codeword indexes $i_x^1, i_x^2, ..., i_x^M$ of item x
2: **Output:** An approximation of $q^\top x$
3: $l = 0, p = 0$;
4: **for** m from 1 to M' **do**
5: $\quad l = l + l^m[i_x^m]$;
6: **end for**
7: **for** m from $M' + 1$ to M **do**
8: $\quad p = p + q^\top c^m[i_x^m]$;
9: **end for**
10: return $l \cdot p$;

The remaining problem is how to train the norm and vector codebooks. A straightforward solution, which trains the norm codebooks with $\|x\|$ and the vector codebooks with $x/\|x\|$, does not work. This is because the codebook based approximation $\bar{x} = \sum_{m=M'+1}^{M} c^m[i_{x'}^m]$ of the direction vector is not guaranteed to be unit norm due to the intrinsic norm errors of vector quantization. Therefore, even if we quantize $\|x\|$ accurately with the norm codebooks, $\tilde{x}$ in (3) can still have large norm error. NEQ solves this problem with the codebook learning process in Algorithm 2.

Algorithm 2 NEQ: Codebook Learning

1: **Input:** Dataset $\mathcal{X}$, # codebook M, # norm codebook M'
2: **Output:** M' norm codebooks, $M - M'$ vector codebooks
3: Extract the direction vector $x' = \frac{x}{\|x\|}$;
4: Train $M - M'$ vector codebooks on x' using a VQ method;
5: Encode x' with the vector codebooks, obtain the codebook based approximation $\bar{x}$ of x';
6: Get the *relative norm* l_x of item x as $\frac{\|x\|}{\|\bar{x}\|}$;
7: Train M' norm codebooks to quantize l_x;
8: Return the M codebooks;

[1] SIFT1M is sampled from the SIFT100M dataset used in the experiments in Section 5.

[2] Researches (Neyshabur and Srebro 2015; Shrivastava and Li 2014; Guo et al. 2016) on MIPS usually use a value of k ranging from 1 to 50, 20 is the middle of this range.

[3] See https://arxiv.org/pdf/1911.04654.pdf for the supplementary material.

Line 4 trains the vector codebooks using an exiting VQ method, such as PQ or RQ. Instead of quantizing the actual norm $\|x\|$, NEQ quantizes the relative norm $l_x = \|x\|/\|\bar{x}\|$ in Line 7 of Algorithm 2. This design absorbs the norm error of VQ into the relative norm l_x and ensures that the codebook based approximation $\tilde{x}$ in (3) has the same norm as x if l_x is quantized accurately. As we will show in the experiments, NEQ also works for datasets in which items have almost identical norms thanks to this design. The norm codebooks are learned in a recursive manner similar to RQ. The norm is used to train the first codebook $\mathcal{L}^1$ with K-means. The residuals ($\|x\| - l^1[i_x^1]$) are used to train $\mathcal{L}^2$ and this process is conducted iteratively. The normalization in Line 3 may look unnecessary as we can quantize the original item x directly using the vector codebooks and define the relative norm as $\|x\|/\|\tilde{x}\|$. However, we observed that this alternative does not perform as well as Algorithm 2. One possible reason is that unit vectors may be easier to quantize for VQ techniques.

As a demonstration of the effectiveness of NEQ in reducing the quantization error in norm, we report some statistics of the Yahoo!Music dataset. For the original RQ, a norm error of 1.51×10^{-2} and 6.47×10^{-3} are achieved with 8 and 16 codebooks, respectively. Keeping the total number of codebooks the same and using only one codebook for norm, norm explicit quantization based RQ reaches a norm error of 1.1×10^{-3} under both 8 and 16 codebooks. We will show that the lower norm error of NEQ translates into better performance for MIPS in the experiments in Section 5.

Setting the number of norm codebooks. Generally, a good M' can be chosen by testing the recall-item performance of all $M - 1$ [4] configurations on a set of sample queries. When the number of codewords in each codebook is 256 (i.e., $K = 256$), we found empirically that using one codebook for norm provides the best performance in most cases. This is because the norm error is already small with one norm codebook. Using more codebooks for norm provides limited reduction in norm error but increases angular error as the number of angular codebooks is reduced.

Why not storing the norm? As the relative norm l_x is a scalar, one may wonder why not storing its exact value to completely eliminate norm error. This is because storing l_x with a 4-byte floating point number costs too much space and VQ algorithms are usually evaluated with a fixed per-item space budget (especially when used for data compression). With the usual setting $K = 256$, using M codebooks results in a per-item index size of M bytes. If l_x is stored exactly, the direction vector can only use $M - 4$ codebooks. Empirically, we found that using 1 norm codebook already makes the norm error very small, which leaves direction vector $M - 1$ codebooks and achieves better overall performance.

Complexity analysis. For index building, NEQ learns $M - M'$ vector codebooks and the original VQ method learns M vector codebooks. Although NEQ needs to conduct normalization twice (Line 3 and Line 6 of Algorithm 2) and learn the norm codebooks, the complexity of these operations is generally low compared with learning vector codebooks. For inner product computation with lookup table, the original VQ method needs M lookups and $M - 1$ additions. NEQ needs M' lookups and $M' - 1$ additions to reconstruct the relative norm, and $M - M'$ lookups and $M - M' - 1$ additions to add the inner product. Then one more multiplication is needed to assemble the final result. Thus, approximate inner product computation in NEQ costs M lookups and $M - 1$ additions, which is exactly the same as the original VQ method. Therefore, NEQ does not increase the complexity of codebook learning and approximate inner product computation.

We would like to emphasize that the strength of NEQ lies in its simplicity and generality. NEQ is simple in that it uses existing VQ methods to quantize the direction vector without modifying their formulations of the codebook learning problem. This makes NEQ easy to implement as off-the-shelf VQ libraries can be reused. NEQ is also general in that it can be combined with any VQ methods, including PQ, OPQ, RQ and AQ. In the supplementary material, we show that NEQ with two codebooks can adopt the multi-index algorithm (Babenko and Lempitsky 2012) for candidate generation in MIPS. We will also show in Section 5 that NEQ boosts the performance of many VQ methods for MIPS.

5 Experiments

Experiment setting. We used four popular datasets, Netflix, Yahoo!Music, ImageNet and SIFT100M, whose statistics are summarized in Table 1. Netflix and Yahoo!Music record user ratings for items. We obtained item and user embeddings from these two datasets using alternating least square (ALS) (Yun et al. 2013) based matrix factorization. The item embeddings were used as dataset items, while the user embeddings were used as queries. ImageNet and SIFT100M contain descriptors of images. The four datasets vary significantly in norm distribution (see details in the supplementary material) and we deliberately chose them to test NEQ's robustness to different norm distributions. ImageNet has a long tail in its norm distribution, while items in SIFT100M have almost the same norm. For Netflix and Yahoo!Music, most items have a norm close to the maximum [5].

Following the standard protocol for evaluating VQ techniques (Babenko and Lempitsky 2014; 2015; Zhang, Du, and Wang 2014), we used the recall-item curve as the main performance metric and it measures the ability of a VQ method to preserve the similarity ranking of the items. To obtain the recall-item curve, all items in a dataset are first sorted according to the codebook based approximate inner products. For a query, denote the set of items ranking top T as $\mathcal{S}'$ and the set of ground truth top-k MIPS results as $\mathcal{S}$, the recall is $|\mathcal{S}' \cap \mathcal{S}|/|\mathcal{S}|$. At each value of T, we report the average recall of 10,000 randomly selected queries. We do not report the running time as the VQ methods have almost identical running time [6] given the same number of codebooks M.

For a VQ method X (e.g., RQ), its NEQ version is denoted as NE-X (e.g., NE-RQ). The NEQ variants use the same num-

[4] There should be at least 1 and at most $M - 1$ norm codebooks.

[5] See https://github.com/xinyandai/product-quantization for all experiment code and data.

[6] AQ and RQ have more expensive inner product table computation than PQ and OPQ. However, this difference has negligible impact on the running time when the dataset is large.

Table 1: Dataset statistics

DATASET	NETFLIX	YAHOO!MUSIC	IMAGENET	SIFT100M
# ITEMS	17,770	136,736	2,340,373	100,000,000
# DIMENSIONS	300	300	150	128

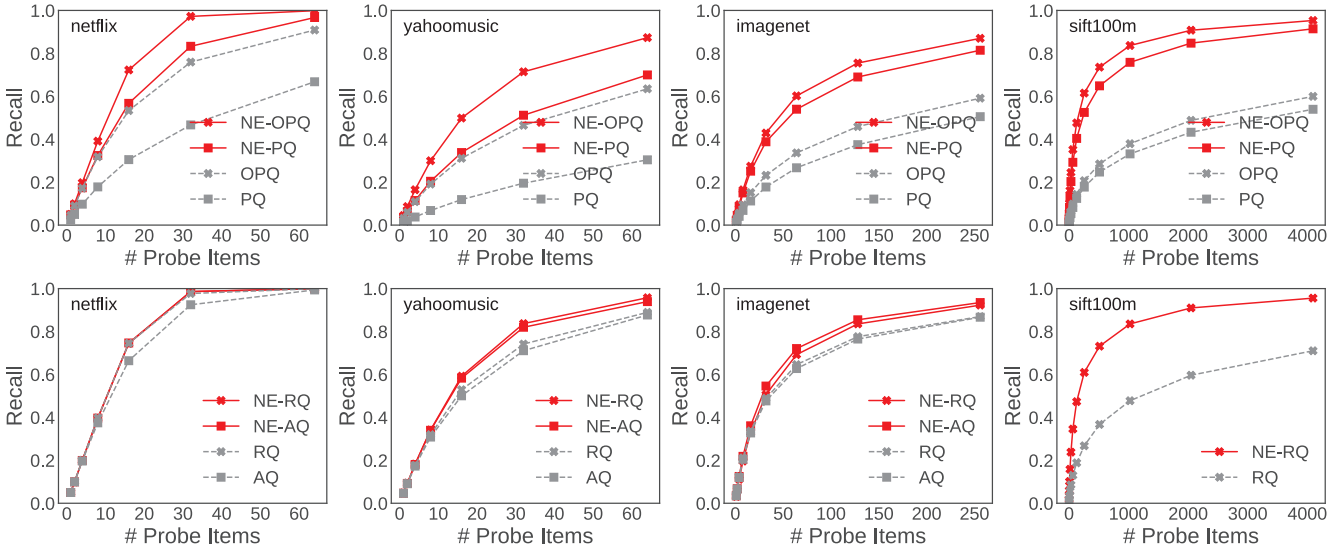

Figure 3: Item-recall performance of the VQ methods and their NEQ-based variants

ber of codebooks (norm codebooks plus direction codebooks) as the original VQ methods. Each codebook has $K = 256$ codewords and only one codebook is used for norm in NEQ. The default value of k (the number of target top inner product items) is 20 [7]. For Neflix, the codebooks were trained using the entire dataset. For the other datasets, the codebooks were trained using a sample of size 100,000.

Improvements over existing VQ methods. We report the performance of the original VQ methods (in dotted lines) and their NEQ-based variants (in solid lines) in Figure 3. The number of codebooks is 8. We do not report the performance of AQ and NE-AQ on SIFT100M as the encoding process of AQ did not finish in 72 hours. The results show that the NEQ-based variants consistently outperform their counterparts on all the four datasets. The performance improvements of NEQ on PQ and OPQ are much more significant than on AQ and RQ. Moreover, there is a trend that the performance benefit increases with the dataset cardinality. These two phenomenons can be explained by the fact that reducing the error in norm is more helpful when the quantization error is large. With 8 codebooks, the small Netflix dataset is already quantized accurately, while the SIFT100M dataset is not well quantized. With the same number of codebooks, PQ and OPQ generally have larger quantization errors than RQ and AQ and thus the performance gain of NEQ is more significant.

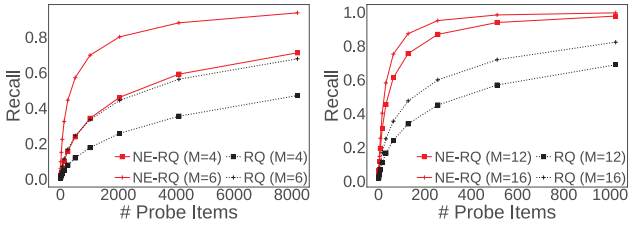

Figure 4: Different number of codebooks

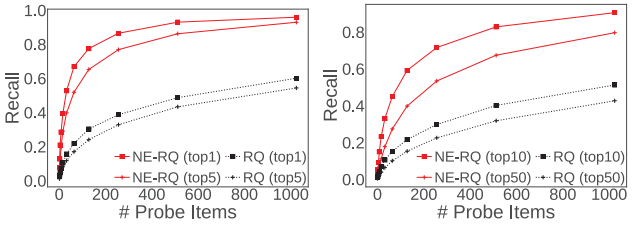

Figure 5: Different values of k

Next, we test the robustness of NEQ to the parameter configurations, i.e., the number of codebooks M and the value of k. We report the performance of RQ and NE-RQ on the SIFT100M dataset in Figure 4 and Figure 5 (the results of other VQ methods and datasets can be found in the supple-

[7] The performance of MIPS is usually evaluated by setting k as 1, 10, 20 or 50 and the results are usually consistent under different configurations of k. Due to space limit, we provide the results under more configurations of k in the supplementary material.

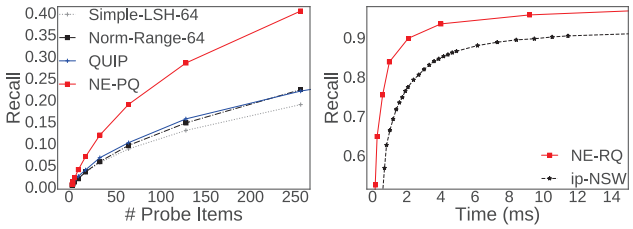

Figure 6: Comparison with LSH & graph methods

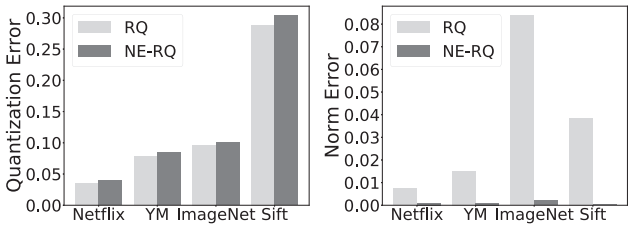

Figure 7: Quantization error of NE-RQ and RQ

mentary material). Figure 4 shows that NE-RQ outperforms RQ across different number of codebooks. Figure 5 shows that NE-RQ consistently outperforms RQ for different values of k with 8 codebooks and the performance gap is similar for different values of k. The results in the supplementary material show that the robustness of NEQ to the parameter configurations also holds for PQ, OPQ and AQ. In addition, we examine the robustness of the VQ methods and their NEQ variants across different runs of the codebook learning algorithm in the supplementary material. The results show that NEQ usually provides smaller standard deviation in recall across different runs.

Comparison with other methods LSH are widely used for similarity search (Li et al. 2018), and Norm-Range LSH (Yan et al. 2018) and Simple-LSH (Neyshabur and Srebro 2015) are the state-of-the-art LSH-based algorithms for MIPS (based on binary hashing). QUIP (Guo et al. 2016) is a vector quantization method specialized for MIPS, which explicitly minimizes the squared inner product error ($(q^\top x - q^\top \tilde{x})^2$) to learn the codebooks. QUIP has several variants and we used *QUIP-cov(x)* for fair comparison as other variants use knowledge about the queries but NEQ does not. According to the QUIP paper, the performance gap between other variants and *QUIP-cov(x)* is small for the ImageNet dataset. For Norm-Range LSH, we partitioned the dataset into 64 sub-datasets as recommended in (Yan et al. 2018). We report the performance results on the ImageNet dataset in Figure 6 (left). Simple-LSH and Norm-Range used 64 bit binary code. NE-PQ and QUIP use two codebooks each containing 256 codewords. This means that the per item index size of NE-PQ (and QUIP) is 16 bit and only a quarter of that of the LSH-based methods. The results show that the vector quantization based methods (NE-PQ and QUIP) outperform the LSH-based algorithms with smaller per-item index size. Moreover, NE-PQ significantly outperforms QUIP even if QUIP uses a more complex codebook learning strategy.

We also compared the recall-time performance of NE-RQ with the proximity graph-based ip-NSW algorithm (Morozov and Babenko 2018) on the ImageNet dataset in Figure 6 (right). ip-NSW is shown to achieve the state of the art recall-time performance in existing MIPS algorithms in (Morozov and Babenko 2018). NE-RQ with two codebooks was used for candidate generation (by combining with the multi-index algorithm (Babenko and Lempitsky 2012)) and the candidates were verified by calculating the exact inner product in this experiment. The results show that NE-RQ achieves higher recall than ip-NSW given the same query processing time. As the implementation may affect the running time, we also plot recall vs. inner product calculation in the supplementary material, which shows that NE-RQ requires fewer inner product computation at the same recall. However, we found ip-NSW provides better recall-time performance than NEQ on the SIFT1M dataset. Although the main design goal of NEQ is good recall-item performance instead of recall-time performance, this experiment shows that using NEQ to generate candidate is beneficial to some datasets.

Insights. A natural question arises after observing the good performance of NEQ: *Does NEQ only reduce the error in norm? Or it reduces the quantization error as a by-product of its design?* To answer this question, we compared the quantization error ($\|x - \tilde{x}\|$ normalized by the maximum norm in the dataset) and the norm error of RQ and NE-RQ in Figure 7. The number of codebooks is 8 and the reported errors are averaged over all items in the dataset. The results show that NE-RQ indeed reduces norm error significantly but its quantization error is slightly larger than RQ on all the four datasets. This can be explained by the fact that NE-RQ uses 1 codebook to encode the norm and has fewer vector codebooks than RQ. This result shows that a smaller quantization error does not necessarily result in better performance for MIPS. Originally designed for Euclidean distance, existing VQ methods minimize the quantization error. With NEQ, we have shown that the minimizing quantization error is not a suitable design principle for inner product due to its unique properties.

6 Conclusions

In this paper, we questioned whether minimizing the quantization error is a suitable design principle of VQ techniques for MIPS. We found that the quantization error in norm have great influence on inner product and can be significantly reduced by explicitly encoding it using separate codebooks. Based on this observation, we proposed NEQ — a general paradigm that specializes existing VQ techniques for MIPS. NEQ is simple as it does not modify the codebook learning process of existing VQ methods. NEQ is also general as it can be easily combined with existing VQ methods. Experimental results show that NEQ provides good performance consistently on various datasets and parameter configurations. Our work shows that inner product requires different design principles from Euclidean distance for VQ techniques and we hope to inspire more researches in this direction.

Acknowledgments. This work was supported by ITF 6904945, and GRF 14208318 & 14222816, and the National

Natural Science Foundation of China (NSFC) (Grant No. 61672552).

References

Babenko, A., and Lempitsky, V. S. 2012. The inverted multi-index. In *CVPR*, 3069–3076.

Babenko, A., and Lempitsky, V. S. 2014. Additive quantization for extreme vector compression. In *CVPR*, 931–938.

Babenko, A., and Lempitsky, V. S. 2015. Tree quantization for large-scale similarity search and classification. In *CVPR*, 4240–4248.

Cai, T. T.; Fan, J.; and Jiang, T. 2013. Distributions of angles in random packing on spheres. *JMLR* 14(1):1837–1864.

Chandar, S.; Ahn, S.; Larochelle, H.; Vincent, P.; Tesauro, G.; and Bengio, Y. 2016. Hierarchical memory networks. *CoRR* abs/1605.07427.

Chen, Y.; Guan, T.; and Wang, C. 2010. Approximate nearest neighbor search by residual vector quantization. *Sensors* 10(12):11259–11273.

Dean, T. L.; Ruzon, M. A.; Segal, M.; Shlens, J.; Vijayanarasimhan, S.; and Yagnik, J. 2013. Fast, accurate detection of 100, 000 object classes on a single machine. In *CVPR*, 1814–1821.

Douze, M.; Sablayrolles, A.; and Jégou, H. 2018. Link and code: Fast indexing with graphs and compact regression codes. In *CVPR*, 3646–3654.

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D. A.; and Ramanan, D. 2010. Object detection with discriminatively trained part-based models. *TPAMI* 32(9):1627–1645.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2946–2953.

Gersho, A., and Gray, R. M. 1991. *Vector quantization and signal compression*. Kluwer.

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

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *TPAMI* 33(1):117–128.

Johnson, J.; Douze, M.; and Jégou, H. 2017. Billion-scale similarity search with gpus. *CoRR* abs/1702.08734.

Jun, K.; Bhargava, A.; Nowak, R. D.; and Willett, R. 2017. Scalable generalized linear bandits: Online computation and hashing. In *NeurIPS*, 99–109.

Kalantidis, Y., and Avrithis, Y. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2329–2336.

Koenigstein, N.; Ram, P.; and Shavitt, Y. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *CIKM*, 535–544.

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

Li, H.; Chan, T. N.; Yiu, M. L.; and Mamoulis, N. 2017. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD*, 835–850.

Li, J.; Yan, X.; Zhang, J.; Xu, A.; Cheng, J.; Liu, J.; Ng, K. K. W.; and Cheng, T. 2018. A general and efficient querying method for learning to hash. In *SIGMOD*, 1333–1347.

Martinez, J.; Clement, J.; Hoos, H. H.; and Little, J. J. 2016. Revisiting additive quantization. In *ECCV*, 137–153.

Morozov, S., and Babenko, A. 2018. Non-metric similarity graphs for maximum inner product search. In *NeurIPS*, 4726–4735.

Mussmann, S., and Ermon, S. 2016. Learning and inference via maximum inner product search. In *ICML*, 2587–2596.

Neyshabur, B., and Srebro, N. 2015. On symmetric and asymmetric lshs for inner product search. In *ICML*, 1926–1934.

Ram, P., and Gray, A. G. 2012. Maximum inner-product search using cone trees. In *SIGKDD*, 931–939.

Shrivastava, A., and Li, P. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *NeurIPS*, 2321–2329.

Teflioudi, C.; Gemulla, R.; and Mykytiuk, O. 2015. LEMP: fast retrieval of large entries in a matrix product. In *SIGMOD*, 107–122.

Wu, X.; Guo, R.; Suresh, A. T.; Kumar, S.; Holtmann-Rice, D. N.; Simcha, D.; and Yu, F. X. 2017. Multiscale quantization for fast similarity search. In *NeurIPS*, 5745–5755.

Yan, X.; Li, J.; Dai, X.; Chen, H.; and Cheng, J. 2018. Norm-ranging LSH for maximum inner product search. In *NeurIPS*, 2956–2965.

Yun, H.; Yu, H.; Hsieh, C.; Vishwanathan, S. V. N.; and Dhillon, I. S. 2013. NOMAD: non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *CoRR* abs/1312.0193.

Zhang, T.; Du, C.; and Wang, J. 2014. Composite quantization for approximate nearest neighbor search. In *ICML*, 838–846.