Online Additive Quantization

Qi Liu
qiliu67@mail.ustc.edu.cn
University of Science and Technology of China

Jin Zhang
abczj@mail.ustc.edu.cn
University of Science and Technology of China

Defu Lian*
liandefu@ustc.edu.cn
University of Science and Technology of China

Yong Ge
yongge@arizona.edu
The University of Arizona

Jianhui Ma
jianhui@ustc.edu.cn
University of Science and Technology of China

Enhong Chen
cheneh@ustc.edu.cn
University of Science and Technology of China

ABSTRACT

Approximate nearest neighbor search (ANNs) plays an important role in many applications ranging from information retrieval, recommender systems to machine translation. Several ANN indexes, such as hashing and quantization, have been designed to update for the evolving database, but there exists a remarkable performance gap between them and retrained indexes on the entire database. To close the gap, we propose an online additive quantization algorithm (online AQ) to dynamically update quantization codebooks with the incoming streaming data. Then we derive the regret bound to theoretically guarantee the performance of the online AQ algorithm. Moreover, to improve the learning efficiency, we develop a randomized block beam search algorithm for assigning each data to the codewords of the codebook. Finally, we extensively evaluate the proposed online AQ algorithm on four real-world datasets, showing that it remarkably outperforms the state-of-the-art baselines.

CCS CONCEPTS

• **Information systems** → **Nearest-neighbor search**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Additive Quantization, Beam Search, Nearest Neighbor Search, Online Update

ACM Reference Format:
Qi Liu, Jin Zhang, Defu Lian*, Yong Ge, Jianhui Ma, and Enhong Chen. 2021. Online Additive Quantization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467441

*Corresponding author: Defu Lian

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467441

1 INTRODUCTION

The advent of Internet has led to massive information overload in recent decades. For example, Google indexes more than 1 trillion webpages, Twitter hosts hundreds of millions of tweets and Flickr has billions of images. One method to address the information overload is to search for relevant information in data oceans with queries, but it turns out to be a challenging task. It is boiled down to the nearest neighbor search problem in a given database [18, 32], which can be efficiently yet approximately solved by the Approximate Nearest Neighbor search (ANNs) techniques, including hashing [33], quantization [23], tree [30] or graph index-based approaches [20, 21]. In addition to efficient information filtering, ANNs also plays an important role in many other tasks, such as recommender systems [24, 26, 28, 31], machine translation [7, 36], and multi-class classification [11, 25].

Due to data generation at an unprecedented rate per day, databases are dynamically growing while the data distribution may evolve over time. For instance, Twitter receives over 100 million tweets per day, Flickr receives over 3,000 images per minute, and Youtube has more than 100 hours of videos uploaded per minute. Without incorporating newly generated data, ANNs may not provide highly accurate responses to achieve satisfactory performance. However, it is almost computationally impractical to train the ANNs method from scratch each time the new data comes in due to the large size of the database. Therefore, it is important to develop ANNs algorithms to handling incremental data with a low computational cost.

Several studies have been conducted to support online learning of ANNs algorithms. There are mainly two lines of directions for this task. One line of research direction is online hash [3–5, 8, 12, 13, 17], to adapt hashing-based ANNs to accommodating the incremental data. The main idea is to update the hash function with the new data and then update the hash codes of existing stored data via the new hash functions. The advantage of the hash-based ANNs methods lies in the low computational cost of search and low storage cost of binary codes. However, the problems of these methods include low accuracy of approximation due to low capacity of representation, and the high computation cost of code maintenance due to the high frequent update of the hash functions. Moreover, these methods require keeping the old data so that the new hash code of the old data can be updated. Another line of research direction is online quantization [6, 35], to update codebooks to incorporate incremental data. The most related work is online product

quantization [35], which leverages online K-means to update the codebooks without recomputing the codes (i.e., indexes) of old data with respect to the updated codebooks, thus reducing the computational cost of learning and maintenance. However, the problem of these methods lies in $O(T)$ regret bounds of online learning*, such that the accuracy of the approximate search would always decay over time. Its variant over a sliding window has been proposed to reduce the regret bound by decreasing T, but the issue remains in the same setting.

To this end, we propose an online learning algorithm for additive quantization, which is called online Additive Quantization (online AQ). AQ performs empirically better than (optimized) product quantization and comparably to composite quantization. Therefore, it is worth investigating how to adapt AQ to accommodate incremental data. When following the same assumption as online PQ to not update the codes of old data (i.e., indexes with respect to codebooks), the codebooks of AQ can be easily and efficiently updated with a closed-form equation to incorporate new data based on the matrix inversion lemma. In this way, we derive a better regret bound than online PQ, which can theoretically guarantee the performance of the proposed online AQ algorithm. Moreover, AQ has developed the beam search algorithm to assign the codewords of codebooks with the smallest quantization error to the data. Due to its cubic complexity with the number of books, AQ suffers from a high computational cost. For the sake of improving efficiency, we proposed a block-wise beam search algorithm over randomly-selected blocks to compute codes for each streaming data.

To summarize, we deliver the following new contributions:

- To the best of our knowledge, we propose the online learning algorithm for additive quantization for the first time, where the codebooks can be efficiently updated to accommodate the new data, and the codes of the new data can be fast computed with the novel randomized block beam search algorithm.
- We derive a better regret bound than online product quantization to guarantee the performance of the proposed online AQ algorithm. The empirical comparison between the decaying curving of quantization error can confirm the superiority of online AQ.
- The online AQ algorithm is comprehensively evaluated on four real-world datasets, where the results demonstrate that online AQ outperforms the state-of-the-art online learning to index methods.

2 RELATED WORKS

Online additive quantization is mainly related to hashing and quantization, therefore we only focus on the advance of hashing and quantization. Though tree and graph-based indexes are also used for approximate nearest neighbor search, few of them have been adapted to the dynamic databases, so these techniques are not discussed here due to the space limits. Interested readers could refer to the survey [18].

2.1 Hashing and Quantization for ANNs

Hashing-based ANNs can be categorized into data-independent hashing and data-dependent hashing. Locality sensitive hash (LSH) is one of the most representative work for data-independent hashing. LSH generates randomized hash functions, which can theoretically guarantee that more similar data can be mapped into the same bucket with a higher probability, where buckets are identified by the hash code of the data. Data-dependent hashing learns hash functions from the data, which are likely to achieve higher performance than the former methods. Semantic hashing was proposed to learn binary codes via Restricted Boltzmann Machine for fast searching similar documents [29]. Spectral hashing applied spectral analysis techniques to embed a constructed similarity graph between data points into a binary Hamming space [34]. ITQ [16] and IsoH [10] proposed to learn a rotation matrix for similar or isotropic variances in projected dimensions to derive more effective and compact binary codes. Please refer to [33] for a more comprehensive survey.

Quantization-based ANNs usually derive multiple codebooks by minimizing the quantization error between data points and the composition of codewords. From each codebook, one and only one codeword is selected. It is possible to composite these codewords by concatenation and addition, such that an exponentially large codebook can be generated. Product quantization [14], as one of the most representative work for quantization, decomposed the vector representation space into the Cartesian product of subspaces and then applied k-means for deriving codebook in each separate subspace. Then, each data point is approximated by concatenating codewords selected from these codebooks. Optimized product quantization [9] jointly learned space decomposition and subspace quantization. Composite Quantization [37] and Additive Quantization [1] do not decompose space but directly learned multiple codebooks in an iterative way. Sparse constraints can be imposed on codebooks, enabling fast encoding and evaluation [2, 38].

2.2 Online Learning for ANNs

Since data-independent hashing methods are not dependent on the input data, they can be easily and straightforwardly adapted to accommodate streaming data. Online data-dependent hashing methods receive much attention in recent years. They can be grouped into supervised online hashing and unsupervised online hashing according to the availability of label information. The representative work of the former taxonomy includes Online Kernel Hashing [12, 13], Adaptive Hash [5], Online Supervised Hashing [3] and Online Hashing with Mutual Information [4]. These methods are distinguished from each other by how to leverage label information under the online supervised learning framework. Due to the lack of label information, the methods of the latter taxonomy only model the inherent properties among data. The representative work includes stream spectral binary coding, Online Sketching Hashing [17] and Faster Online Sketching Hashing [8]. The main idea of these methods is to exploit matrix sketching techniques [19] for approximating the large data matrix with another smaller matrix and then to learn hash functions based on the smaller matrix. Since quantization-based methods only lead to lower quantization errors than hashing, online product quantization was proposed to adapt product quantization to accommodate the streaming data [35]. Online PQ applies online K-means for updating the codebooks at the assumption of not recomputing the index of old

*T is the number of time steps for online learning

data. In spite of this rigid assumption, online PQ can be efficiently learned and is remarkably superior to the hashing-based methods. The proposed online AQ is unsupervised quantization, which does not leverage any label information.

3 PRELIMINARY

Assume online AQ operates the streaming data, which are organized as a large matrix $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]^\top \in \mathbb{R}^{T\times D}$. Each row of the matrix Y is received one after the other in a streaming fashion. We first introduce basic knowledge in the static setting, where the whole matrix Y is given and then extend to the dynamic setting. The most basic vector quantization is to quantize a vector $\boldsymbol{y}_t$ to a codeword $\boldsymbol{w}_k$ in a codebook $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_K]$, such that the quantization distortion $\|\boldsymbol{y}_t - \boldsymbol{w}_k\|$ between them is minimized. The additive quantization is to quantize the vector $\boldsymbol{y_t}$ to the addition of codewords, each of which is selected from a codebook. The additive quantization assumes to use M codebooks, $\boldsymbol{W}^{(1)}, \cdots, \boldsymbol{W}^{(M)}$, and approximate the data $\boldsymbol{y}_t$ with $\hat{\boldsymbol{y}}_t$, which is defined as follows:

$$\hat{\boldsymbol{y}}_t = \sum_{m=1}^{M} \boldsymbol{w}^{(m)}_{i_m(\boldsymbol{y}_t)} \tag{1}$$

where $i_m(\boldsymbol{y}_t) \in \{1, \ldots, K\}$ returns the index in the m-th codebook of the data $\boldsymbol{y}_t$. We can represent each index as a one-hot vector, and concatenate these one-hot vectors in a specified order to obtain the index vector $\boldsymbol{x}_t = [\boldsymbol{x}_t^{(1)}, \cdots, \boldsymbol{x}_t^{(M)}] \in \{0,1\}^{MK\times 1}$, where the k-th element $x_{t,k}^{(m)}$ of the vector $\boldsymbol{x}_t^{(m)}$ equals to 1 if $i_m(\boldsymbol{y}_t) = k$ and 0 otherwise. If we further concatenate the codebook matrix $\boldsymbol{W}^{(m)}$ by column to obtain the matrix $\boldsymbol{W} = [\boldsymbol{W}^{(1)}, \cdots, \boldsymbol{W}^{(M)}] \in \mathbb{R}^{MK\times D}$, the approximated vector can be reformulated as

$$\hat{\boldsymbol{y}}_t = \boldsymbol{W}^\top \boldsymbol{x}_t. \tag{2}$$

Then, to quantize the data $\boldsymbol{y}_t$, the additive quantization finds the best $\boldsymbol{x}_t$ with respect to $\boldsymbol{W}$ by optimizing

$$\boldsymbol{x}_t^*(\boldsymbol{W}) = \arg\min_{\boldsymbol{x}_t} \|\boldsymbol{y}_t - \boldsymbol{W}^\top \boldsymbol{x}_t\|_2^2. \tag{3}$$

According to the definition of quantization error, it equals $\|\boldsymbol{y}_t - \boldsymbol{W}^\top \boldsymbol{x}_t^*\|_2$ for AQ. However, it is easy to observe that this involves a combinatorial optimization problem, which suffers from high computational cost. Therefore, AQ developed a beam search algorithm for finding the approximate solution.

The codebook of AQ is usually learned with the goal of minimizing the quantization error of the whole dataset

$$\boldsymbol{W}^*(X) = \arg\min_{\boldsymbol{W}} \sum_{t=1}^{T} \|\boldsymbol{y}_t - \boldsymbol{W}^\top \boldsymbol{x}_t\|_2^2 = \|Y - X\boldsymbol{W}\|_F^2 \tag{4}$$

where the matrix Y, of size $T \times D$, is obtained via stacking $\boldsymbol{y}_t$ by rows, and the matrix $\boldsymbol{X}$, of size $T \times MK$, is obtained via stacking $\boldsymbol{x}_t$ by rows. This is a linear regression problem, which has a closed-form solution. However, to make the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ invertable, we introduce a regularizer term $\lambda\|\boldsymbol{W}\|_F^2$, so that the closed-form solution is given as follows:

$$\boldsymbol{W}^*(X) = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top Y \tag{5}$$

Assume $\mathrm{C}_k^{(m)} = \{t \in \{1, \cdots, T\} | i_m(\boldsymbol{y}_t) = k\}$ be the set of data indexes assigned to the k-th codeword in the m-th codebook. Then the element $[\boldsymbol{X}^\top \boldsymbol{X}]_{m,m',k,k'} = |\mathrm{C}_k^{(m)} \cap \mathrm{C}_{k'}^{(m')}|$. As long as there is an empty index set, the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is not full rank and thus not invertable. When K is large, it is highly likely that there are some empty index set. Therefore, it is necessary to impose the Frobenius norm regularizer.

Table 1: Notations and Definitions

Notation	Definition
t	the step index. $t = 1, 2, \ldots, T$
M	the number of codebooks
K	the number of codewords in each codebook
n_0	the number of vectors in the initial database
$\boldsymbol{y}_t \in \mathbb{R}^D$	the new data at step t
$y_{t,k} \in \mathbb{R}$	the k-th entry of $\boldsymbol{y}_t$
$\boldsymbol{x}_t \in \{0,1\}^{MK\times 1}$	the index vector of $\boldsymbol{y}_t$ w.r.t codewords $\|\boldsymbol{x}_t\|_1 = M$
$\boldsymbol{x}_{0,i} \in \{0,1\}^{MK\times 1}$	the i-th index vector in the initial database
$\boldsymbol{A}_0 \in R^{MK\times MK}$	$\boldsymbol{A}_0 = \sum_{i=1}^{n_0} \boldsymbol{x}_{0,i}\boldsymbol{x}_{0,i}^\top + \lambda \boldsymbol{I}$
$\boldsymbol{A}_t \in R^{MK\times MK}$	$\boldsymbol{A}_t = \boldsymbol{A}_0 + \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^\top$
$\boldsymbol{W}_t \in R^{MK\times D}$	codebook matrix of all codewords at step t
$\boldsymbol{w}_{t,k} \in R^{MK\times 1}$	The k-th column of the matrix $\boldsymbol{W}_t$
$\|\cdot\|_t$	$\|\boldsymbol{u}\|_t = \sqrt{\boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u}}$

4 ONLINE ADDITIVE QUANTIZATION

In this section, we first introduce how to adapt additive quantization to the dynamic database with streaming data, and then discuss how to speed up the computation of beam search.

4.1 Accommodating Streaming Data

Assume we have n_0 data points available in the beginning, where the i-th data is represented by $\boldsymbol{y}_{0,i}$. These data points are stacked by rows, forming an initial data matrix Y_0. We first run AQ on the initial data set, and obtain the initial codebooks $\boldsymbol{W}_0$ and initial the index vector $\boldsymbol{x}_{0,i}$ for each data point $\boldsymbol{y}_{0,i}$. Stacking $\boldsymbol{x}_{0,i}$ by row derive index matrix $\boldsymbol{X}_0$. Then in the t-step, the incoming data is $\boldsymbol{y}_t$, we should first find the index vector $\boldsymbol{x}_t$ using the beam search algorithm and then update the codebooks to reflect the addition of the data point $\boldsymbol{y}_t$. Similar to the online PQ, we also assume the index of the old data does not recompute when the codebooks are updated with respect to the new data. Then the codebooks can be updated with a closed-form equation based on the Sherman–Morrison formula. More concretely, letting $\boldsymbol{W}_t$ the codebook at the step t, $\boldsymbol{A}_t = \boldsymbol{X}_0^\top \boldsymbol{X}_0 + \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^\top + \lambda \boldsymbol{I}$, $\boldsymbol{R}_t = \boldsymbol{X}_0^\top Y_0 + \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{y}_i^\top$,

Algorithm 1: Online Additive Quantization

$[\boldsymbol{W}_0, \boldsymbol{X}_0] \leftarrow \text{AQ}(\boldsymbol{Y}_0, M, K)$;
$\boldsymbol{A}_0^{-1} \leftarrow (\boldsymbol{X}_0^\top \boldsymbol{X}_0 + \lambda \boldsymbol{I})^{-1}$;
for $t \leftarrow 1$ **to** T **do**
  $\boldsymbol{Y}_t \leftarrow$receive the t-batch data ;
  for $i \leftarrow 1$ **to** n_t **do**
    $\boldsymbol{x}_t \leftarrow$beam_search$(\boldsymbol{y}_t, \boldsymbol{W}_{t-1})$;
  $\boldsymbol{A}_t^{-1} \leftarrow \boldsymbol{A}_{t-1}^{-1} - \boldsymbol{A}_{t-1}^{-1}\boldsymbol{X}_t(\boldsymbol{I} + \boldsymbol{X}_t^\top \boldsymbol{A}_{t-1}^{-1}\boldsymbol{X}_t)^{-1}\boldsymbol{X}_t\boldsymbol{A}_{t-1}^{-1}$;
  $\boldsymbol{W}_t \leftarrow$ update codebooks according to Eq (7);

the codebook $\boldsymbol{W}_t$ can be updated as follows:

$$\begin{aligned}\boldsymbol{W}_t &= \boldsymbol{A}_t^{-1}\boldsymbol{R}_t \\ &= (\boldsymbol{A}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t)^{-1}(\boldsymbol{R}_{t-1} + \boldsymbol{x}_t \boldsymbol{y}_t^\top) \\ &= (\boldsymbol{A}_{t-1}^{-1} - \frac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t\boldsymbol{x}_t^\top\boldsymbol{A}_{t-1}^{-1}}{1 + \boldsymbol{x}_t^\top\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t})(\boldsymbol{R}_{t-1} + \boldsymbol{x}_t\boldsymbol{y}_t^\top) \\ &= \boldsymbol{W}_{t-1} + \frac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t(\boldsymbol{y}_t - \boldsymbol{W}_{t-1}^\top\boldsymbol{x}_t)^\top}{1 + \boldsymbol{x}_t^\top\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}.\end{aligned} \tag{6}$$

Therefore, the update of $\boldsymbol{W}_t$ only depends on $\boldsymbol{W}_{t-1}$ and $\boldsymbol{A}_{t-1}^{-1}$, such that the old data are not required to maintain. Besides, since $n_0 \gg 1$, $\boldsymbol{W}_t$ does not change much after adding the new data so that we do not recompute the index for the new data.

4.2 Mini-Batch Extension

In addition to accommodating one streaming data once a time, online AQ is easily extended to incorporate a mini-batch of data. Assuming each time we receive a new batch of data $\boldsymbol{Y}_t \in \mathbb{R}^{n_t \times D}$, the index vector of each data point in this batch can be obtained via beam search. These index vectors are stacked by rows, forming the matrix $\boldsymbol{X}_t$. Similar to the single data case, letting $\boldsymbol{A}_t = \sum_{s=0}^{t}\sum_{i=1}^{n_s}\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top + \lambda\boldsymbol{I}$, and $\boldsymbol{R}_t = \sum_{s=1}^{t}\sum_{i=1}^{n_t}\boldsymbol{x}_{s,i}\boldsymbol{y}_{s,i}^\top$, the codebooks $\boldsymbol{W}_t$ can be updated based on the matrix inversion lemma:

$$\begin{aligned}\boldsymbol{W}_t &= \boldsymbol{A}_t^{-1}\boldsymbol{R}_t \\ &= (\boldsymbol{A}_{t-1} + \boldsymbol{X}_t^\top\boldsymbol{X}_t)^{-1}(\boldsymbol{R}_{t-1} + \boldsymbol{X}_t\boldsymbol{Y}_t^\top) \\ &= \boldsymbol{W}_{t-1} + \boldsymbol{A}_{t-1}^{-1}\boldsymbol{X}_t(\boldsymbol{I} + \boldsymbol{X}_t^\top\boldsymbol{A}_{t-1}^{-1}\boldsymbol{X}_t)^{-1}(\boldsymbol{Y}_t - \boldsymbol{W}_{t-1}^\top\boldsymbol{X}_t)^\top.\end{aligned} \tag{7}$$

For the sake of higher efficiency, we also make the assumption that the number of incremental data is much smaller than the size of the initial database, i.e., $n_0 \gg n_t, \forall t \in \{1, \cdots, T\}$, thus we do also not recompute index for the new batch of data anymore. The overall procedure of online AQ is shown in Algorithm 1, which also subsumes the single data case when the number of the t-step batch data equals 1.

4.3 Randomized Block Beam Search

As aforementioned, AQ exploits beam search for approximate combinatorial optimization in selecting the codewords for the data points. The basic idea of beam search is to consecutively explore codebooks by expanding the most promising codewords in a limited set. To be more specific, considering to select codewords for the vector $\boldsymbol{y}_t$ at the step t, in the beginning, we select the L closest codewords to the vector $\boldsymbol{y}_t$ from $M \times K$ codewords. In the m-th ($m > 1$) step of the following $M-1$ steps, for each candidate tuple with $m-1$ codewords from $m-1$ codebooks, we select the L best codewords in the remaining $M - m + 1$ codebooks, which are the closest to the residual of subtracting the $m-1$ codewords. This will result in L^2 candidate tuples with m codewords, from which the L best candidate tuples with minimal quantization distortion will be kept for the next step.

Algorithm 2: Randomized Block Beam Search

Input: Data point $\boldsymbol{y}$, codebook $\boldsymbol{W}$, the number of groups F
Output: The index vector $\boldsymbol{x}$

$\boldsymbol{x} \leftarrow$ initialize index vector for $\boldsymbol{y}$;
for *iter* $\leftarrow 1$ **to** *#iter* **do**
  $G \leftarrow$ sampling F integers from $\{1 \cdots M\}$ without repl.;
  $\widehat{G} \leftarrow \{1 \cdots M\} - G$;
  $\boldsymbol{W}_G \leftarrow$ extract codebooks with G;
  $\boldsymbol{W}_{\widehat{G}} \leftarrow$ extract codebooks with $\widehat{G}$;
  $\boldsymbol{x}_{\widehat{G}} \leftarrow$ extract index vector with $\widehat{G}$;
  $\tilde{\boldsymbol{y}} \leftarrow \boldsymbol{y} - \boldsymbol{W}_{\widehat{G}}^\top \boldsymbol{x}_{\widehat{G}}$; `// computing residual vector`
  $\boldsymbol{x}_G \leftarrow$ beam_search$(\tilde{\boldsymbol{y}}, \boldsymbol{W}_G)$;

However, the time complexity of the beam search grows cubically with the number of codebooks when the inner product between codebooks are precomputed, so it is not quite impractical to directly apply them in the online learning setting, where codebooks are continuously updated. Another algorithm for solving Eq (3) is hill climbing or iterated conditional modes [22, 37], which iteratively maximizes the objective function with respect to each codeword of a codebook given the assignment among the rest codebooks. This algorithm could be much faster but can not return high-quality solutions.

To give full play to the advantages of both algorithms, we propose a randomized block beam search algorithm. In this algorithm, we iteratively select a group of codebooks in a random way and exploit beam search to assign the codewords out of these codebooks to the data simultaneously when the assignments among the rest codebooks are fixed. If the group size equals 1, this algorithm degenerates to hill climbing. If the group size equals the total number of codebooks, this algorithm becomes the beam search method. The overall procedure is illustrated in Algorithm 2, where we initialize the index vector of the input data with a simple variant of beam search. More concretely, we start to select the L closest codewords to the vector $\boldsymbol{y}$ in the first codebook. In the m-th ($1 < m \leq M$) step, for each candidate tuple, we select the L closest codewords in the m-th codebook to the residual of subtracting the $m-1$ selected codewords. Then the L best candidate tuples of minimal quantization distortion are selected among L^2 candidate tuples. It is also possible to leverage hill climbing for fast initialization, but it does not provide initialize index vector as well as the variant.

4.4 Complexity Analysis

Beam search is replaced with the randomized block beam search in the online additive quantization algorithm, whose time complexity is $O(\#iter F^2 LKD + MLKD)$. The latter part is due to the use of

the simple variant of beam search for initialization, which is M times faster than beam search. Note that the complexity analysis results are different from beam search in Additive Quantization, since it is no more efficient to precompute the inner product between codebooks due to the continuous update of codebooks. The update of codebook and the matrix inverse $\boldsymbol{A}_t^{-1}$ requires $O(n_t^3 + n_t^2MK + n_tM^2K^2 + n_tMKD)$ time complexity. It may be possible to leverage incremental QR decomposition for reducing this cost, but it will be left for future work.

5 THEORETICAL RESULTS

In this section, we study the regret bound for our Online Additive Quantization and assume our framework processes streaming data one at a time. The data point $\boldsymbol{y}_t$ is approximated based on the current codebook $\boldsymbol{W}_t$ with a linear combination of codewords, where the corresponding coefficient vector is $\boldsymbol{x}_t \in \{0,1\}^{MK\times 1}$ with $\|\boldsymbol{x}\|^2 = M$. The codebook is updated by regressing $\boldsymbol{x}_t$ through $y_{t,k}$ on each component of the vector $\boldsymbol{y}_t$. We assume that the optimal coefficient vector of $\boldsymbol{y}_t$ is $\boldsymbol{x}_t^*$, such that $\inf_U \sum_t (\boldsymbol{y_t} - \boldsymbol{U}^\top \boldsymbol{x}_t^*)^2$ is minimized. The regret of online additive quantization is defined as

$$Regret = \sum_{t=1}^{T} \left(\boldsymbol{y}_t - \boldsymbol{W}_t^\top \boldsymbol{x}_t\right)^2 - \inf_{\boldsymbol{U}} \sum_{t=1}^{T} \left(\boldsymbol{y}_t - \boldsymbol{U}^\top \boldsymbol{x}_t^*\right)^2.$$

It is difficult to directly analyze the upper bound of this regret due to non-convexity of the objective function and discreteness of index codes. However, online AQ algorithm has many similarities with online regression, which has developed rich theoretical results about the regret bound. We only need to analyze the gap between our algorithm and the online regression so as to derive our results. The existing results we borrow about the regret bound is based on online regression with bounded loss, that is, the gap between the predicted value $\hat{y}_{t,k}$ and the true value $y_{t,k}$ is bounded by L,

$$|y_{t,k} - \hat{y}_{t,k}| \le L, \forall 1 \le t \le T, 1 \le k \le D.$$

Moreover, the bounded loss of the online regression is optimized by the online mirror descent (OMD) algorithm [27]. In the next part, we will review this background.

5.1 Background

Online Mirror Descent, as shown in Algorithm 3, is run with a sequence of strongly convex functions, $f_0, f_1, ..., f_T$ defined on a common convex domain $S \subseteq X$ and updates model parameter with the gradient of Fenchel Conjugate $f_0^*, f_1^*, ..., f_T^*$ of these functions. See Appendix A for the detailed definitions of Fenchel conjugate and its gradient. More related information can be referred to in the paper [27].

To bridge the gap between online AQ and OMD in this setting, we set $f_t(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u}$, where $\boldsymbol{A}_t = \boldsymbol{A}_{t-1} + \boldsymbol{x}_t\boldsymbol{x}_t^\top$, $\boldsymbol{A}_0 = \sum_{i=1}^{n_0} \boldsymbol{x}_{0,i}\boldsymbol{x}_{0,i}^\top + \lambda \boldsymbol{I}$. It is easy to verify that f_t is 1-strongly convex with respect to the norm $\|\boldsymbol{u}\|_t = \sqrt{\boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u}}$. Its Fenchel Conjugate can be easily derived based on the definition i.e. $f_t^*(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}_t^{-1}\boldsymbol{\theta}$. For $t \ge 1$, letting $\boldsymbol{z}_{t,k} = \boldsymbol{x}_t y_{t,k}$, $\boldsymbol{\theta}_{t+1,k} = \boldsymbol{\theta}_{t,k} + \boldsymbol{z}_{t,k}$ and $\boldsymbol{\theta}_{1,k} = \sum_{i=1}^{n_0} \boldsymbol{x}_{0,i} y_{0,i,k}$, replacing $\boldsymbol{\theta}_{t,k}$ for $\boldsymbol{\theta}_t$ in OMD, and $\boldsymbol{z}_{t,k}$ for $\boldsymbol{z}_t$ leads to the following OMD update $\tilde{\boldsymbol{w}}_{t,k}(\text{OMD}) = \nabla f_t^*\left(\boldsymbol{\theta}_{t,k}\right) = \boldsymbol{A}_t^{-1}\boldsymbol{\theta}_{t,k}$. More concretely,

$$\tilde{\boldsymbol{w}}_{t,k}(\text{OMD}) = \left(\boldsymbol{A}_0 + \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)^{-1} \left(\boldsymbol{\theta}_{1,k} + \sum_{i=1}^{t-1} \boldsymbol{x}_i y_{i,k}\right).$$

Compared to the update in our algorithm, i.e.

$$\boldsymbol{w}_{t,k}(\text{online AQ}) = \left(\boldsymbol{A}_0 + \sum_{i=1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)^{-1} \left(\boldsymbol{\theta}_{1,k} + \sum_{i=1}^{t} \boldsymbol{x}_i y_{i,k}\right),$$

it is easy to observe that

$$\tilde{\boldsymbol{w}}_{t,k} = \boldsymbol{w}_{t,k} - \boldsymbol{A}_t^{-1}\boldsymbol{x}_t y_{t,k},$$

where $\boldsymbol{w}_{t,k} := \boldsymbol{w}_{t,k}$ (online AQ), $\tilde{w}_{t,k} := \tilde{w}_{t,k}(OMD)$.

After establishing the connection, we borrow the theoretical results for deriving our results about regret bound. In particular, based on Lemma 1 in the paper[27], we can derive the following lemma:

Lemma 1. *Assume OMD is run with functions $f_0, f_1, \ldots, f_T$ defined on a common convex domain $S \subseteq X$ and such that each f_t is β_t-strongly convex with respect to the norm $\|\cdot\|_t$, Let $\|\cdot\|_{t,*}$ be the dual norm of $\|\cdot\|_t$, for $t = 1, 2, \ldots, T$. Then, for any $\boldsymbol{u} \in$ S*

$$\sum_{t=1}^{T} \langle \boldsymbol{z}_t, \boldsymbol{u} - \boldsymbol{w}_t \rangle \le \sum_{t=1}^{T} \left(\frac{\|\boldsymbol{z}_t\|_{t,*}^2}{2\beta_t} + f_t^*\left(\boldsymbol{\theta}_t\right) - f_{t-1}^*\left(\boldsymbol{\theta}_t\right)\right) + f_T(\boldsymbol{u}) + f_0^*\left(\boldsymbol{\theta}_1\right).$$

Moreover, for all $t \ge 1$, we have

$$f_t^*\left(\boldsymbol{\theta}_t\right) - f_{t-1}^*\left(\boldsymbol{\theta}_t\right) \le f_{t-1}\left(\boldsymbol{w}_t\right) - f_t\left(\boldsymbol{w}_t\right).$$

The proof is provided in the Appendix.

5.2 Regret Bound

Based on Lemma 1, we can derive the following theorem about regret bounds of online additive quantization.

Theorem 1. *Assume $\forall 1 \le t \le T$, $1 \le k \le D$, $\left|y_{t,k}\right| \le Y$, $\left|y_{t,k} - w_{t,k}x_t\right| \le L$, $\left|y_{t,k} - \tilde{\boldsymbol{w}}_{t,k}^\top \boldsymbol{x}_t\right| \le L$, $\max_k f_0^*\left(\boldsymbol{\theta}_{1,\mathrm{k}}\right) = F$, $\|\boldsymbol{u}\|^2 \le R$. Besides,assume $\left|\left(y_{t,k} - \boldsymbol{u}^\top \boldsymbol{x}_t\right)^2 - \left(y_{t,k} - \boldsymbol{u}^\top \boldsymbol{x}_t^*\right)^2\right| \le \beta$. Let τ be the times in T that $\boldsymbol{x}_t$ does not match $\boldsymbol{x}_t^*$ i.e. $\|\boldsymbol{x}_t - \boldsymbol{x}_t^*\| \ne 0$. The regret of online AQ is bounded from above by*

$$Regret \le DY(Y+2L)\mathrm{MK}\cdot\ln\left(\frac{n_0+T}{K} + \lambda\right) - DY(Y+2L)\cdot\ln\left(\det\left(\boldsymbol{A}_0\right)\right) + \tau D\beta + D\left\|\boldsymbol{A}_0\right\|_F R + 2DF.$$

Proof. Let's consider the regret in the k dimension and divide it into three parts for estimation

$$\begin{aligned}
R_{T,k}(\boldsymbol{u}) &= \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 - \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \boldsymbol{u}^{\top}\boldsymbol{x}_{\mathrm{t}}^{*}\right)^2 \\
&= \sum_{t=1}^{\mathrm{T}} \left[\left(y_{t,k} - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 - \left(y_{t,k} - \widetilde{\boldsymbol{w}}_{\mathrm{t},k}^{\top}\boldsymbol{x}_t\right)^2\right] \\
&\quad + \sum_{t=1}^{\mathrm{T}} \left[\left(y_{t,k} - \widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 - \left(y_{t,k} - \boldsymbol{u}^{\top}\boldsymbol{x}_t\right)^2\right] \\
&\quad + \sum_{t=1}^{\mathrm{T}} \left[\left(y_{t,k} - \boldsymbol{u}^{\top}\boldsymbol{x}_t\right)^2 - \left(y_{t,k} - \boldsymbol{u}^{\top}\boldsymbol{x}_t^{*}\right)^2\right] \\
&= R_{T,k}^{1}(\boldsymbol{u}) + R_{T,k}^{2}(\boldsymbol{u}) + R_{T,k}^{3}(\boldsymbol{u}).
\end{aligned}$$

We deal with these three parts separately,

$$\begin{aligned}
R_{T,k}^{1}(\boldsymbol{u}) &= \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 - \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 \\
&= \sum_{t=1}^{\mathrm{T}} \left(\widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t\right)\left(y_{t,k} - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t + y_{t,k} - \widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right) \\
&\leq 2LY \cdot \sum_{t=1}^{\mathrm{T}} \boldsymbol{x}_t^{\top}\boldsymbol{A}_t^{-1}\boldsymbol{x}_t,
\end{aligned}$$

since $\tilde{\boldsymbol{w}}_{t,k} = \boldsymbol{w}_{t,k} - \boldsymbol{A}_t^{-1}\boldsymbol{x}_t y_{t,k}$, and $|y_{t,k} - \boldsymbol{w}_{t,k}^{\top}\boldsymbol{x}_t| \leq L$, $|y_{t,k} - \widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t| \leq L, |y_{t,k}| \leq Y$.

Next, we use Lemma 1 to calculate the second term:

$$\begin{aligned}
R_{T,k}^{2}(\boldsymbol{u}) &= \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 - \sum_{t=1}^{\mathrm{T}} \left(y_{t,k} - \boldsymbol{u}^{\top}\boldsymbol{x}_t\right)^2 \\
&= 2\left(\sum_{t=1}^{\mathrm{T}} \left\langle \boldsymbol{x}_t y_{t,k}, \boldsymbol{u} - \widetilde{\boldsymbol{w}}_{t,k}\right\rangle - f_T(\boldsymbol{u})\right) + \boldsymbol{u}^{\top}\boldsymbol{A}_0\boldsymbol{u} \\
&\quad + \sum_{t=1}^{\mathrm{T}} \left(\widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 \\
&\leq 2f_T(\boldsymbol{u}) + 2\sum_{t=1}^{\mathrm{T}} \left(\frac{y_{t,k}^2 ||\boldsymbol{x}_t||_{t,*}^2}{2} + f_t^{*}\left(\boldsymbol{\theta}_{t,k}\right) - f_{t-1}^{*}\left(\boldsymbol{\theta}_{t,k}\right)\right) \\
&\quad + 2f_0^{*}\left(\boldsymbol{\theta}_{1,\mathrm{k}}\right) - 2f_T(\boldsymbol{u}) + ||\boldsymbol{A}_0||_F||\boldsymbol{u}||^2 + \sum_{t=1}^{\mathrm{T}} \left(\widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2 \\
&\leq Y^2 \sum_{t=1}^{\mathrm{T}} \boldsymbol{x}_t^{\top}\boldsymbol{A}_t^{-1}\boldsymbol{x}_t + ||\boldsymbol{A}_0||_F R + 2F,
\end{aligned}$$

since f_t is 1-strongly convex with respect to the norm $\|\boldsymbol{u}\|_t = \sqrt{\boldsymbol{u}^{\top}\boldsymbol{A}_t\boldsymbol{u}}$. For the last inequality, we use $\|x_t\|_{t,*}^2 = \boldsymbol{x}_t^{\top}\boldsymbol{A}_t^{-1}\boldsymbol{x}_t$ and the second part of Lemma 1 i.e. $f_t^{*}\left(\boldsymbol{\theta}_{t,k}\right) - f_{t-1}^{*}\left(\boldsymbol{\theta}_{t,k}\right) \leq f_{t-1}\left(\widetilde{\boldsymbol{w}}_{t,k}\right) - f_t\left(\widetilde{\boldsymbol{w}}_{t,k}\right) = -\frac{1}{2}\left(\widetilde{\boldsymbol{w}}_{t,k}^{\top}\boldsymbol{x}_t\right)^2$.

Based on Lemma 2, which is provided and proved in the appendix, we have

$$\begin{aligned}
\sum_{t=1}^{\mathrm{T}} x_{\mathrm{t}}^{\top}\boldsymbol{A}_t^{-1}\boldsymbol{x}_t &= \sum_{t=1}^{\mathrm{T}} \left(1 - \frac{\det\left(\boldsymbol{A}_{t-1}\right)}{\det\left(\boldsymbol{A}_{\mathrm{t}}\right)}\right) \\
&\leq \ln \frac{\det\left(\boldsymbol{A}_T\right)}{\det\left(\boldsymbol{A}_0\right)} \quad (\text{ because } 1 - x \leq -\ln x, x > 0) \\
&= \ln \frac{\prod_{i=1}^{MK} \lambda_i}{\det\left(\boldsymbol{A}_0\right)} \quad (\lambda_i \text{ are the eigenvalues of the } \boldsymbol{A}_{\mathrm{T}}) \\
&\leq \ln\left(\left(\frac{n_0 + T}{K} + \lambda\right)^{MK}\right) - \ln\left(\det\left(\boldsymbol{A}_0\right)\right).
\end{aligned}$$

For the last inequality, we use AM–GM inequality $\prod_{i=1}^{MK} \lambda_i \leq \left(\frac{\sum \lambda_i}{MK}\right)^{MK}$, $\lambda_i > 0$ and $\sum_i \lambda_i = \mathrm{trace}\left(\boldsymbol{A}_T\right) = (n_0 + T)M + \lambda MK$.

Then, combining $R_{T,k}^{3} \leq \beta\tau$, $Regret = \sum_{k=1}^{D} \sup_{\boldsymbol{u}} R_{T,k}(\boldsymbol{u})$ and simplifying, we obtain the relative regret bound. □

6 EXPERIMENT

In this section, we conduct experiments on four real-world datasets to verify the effectiveness of our online additive quantization algorithm. Firstly, we will introduce the four datasets used in our experiments and metrics for evaluation. After presenting the settings of the experiments, we will compare online AQ with baselines and study the effectiveness of codebook update. Following that, we investigate the efficiency and effectiveness of the randomized block beam search. More experiments about sensitivity analysis, online versus mini-batch, and convergence are shown in the appendix.

6.1 Datasets and Evaluation Metrics

The four image datasets will be used for evaluating the online additive quantization algorithm, and these datasets are also used in online PQ for evaluation. The first dataset, Caltech101, contains 101 categories, a total of 9144 images. The second dataset, Halfdome, includes 107732 images which come from Photo Tourism. The third dataset, Sun397, includes about 108K images which come from 397 different scenes. The last dataset, Cifar10, has 60K 32 x 32 images and each belongs to one of 10 classes. Following online PQ, we extract a 512-d GIST feature vector from each image.

Following online PQ, we also use Recall@K as the quantitative metric for comparing different algorithms and set K to 20. This metric measures the portion of the true nearest neighbors being included in the top-K results.

6.2 Settings

Note that the datasets consist of different classes. In order to simulate the real scenario of streaming data, we stream images by classes, collect images as mini-batches, such that each pair of two consecutive mini-batches include half of the images from the same class and the first mini-batch includes more data than the other mini-batches. For all datasets, we use a dynamic query set in each iteration. Specifically, we use the first mini-batch to obtain the initial codebooks using AQ. After that, when a new batch comes in, we first use the new batch as a query set to do the approximate nearest neighbors search in the existing data for the computing

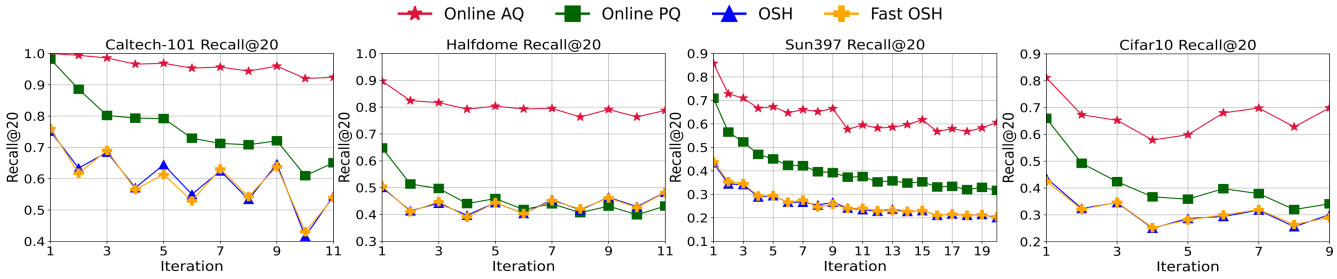

Figure 1: The comparison with the state-of-the-art baselines

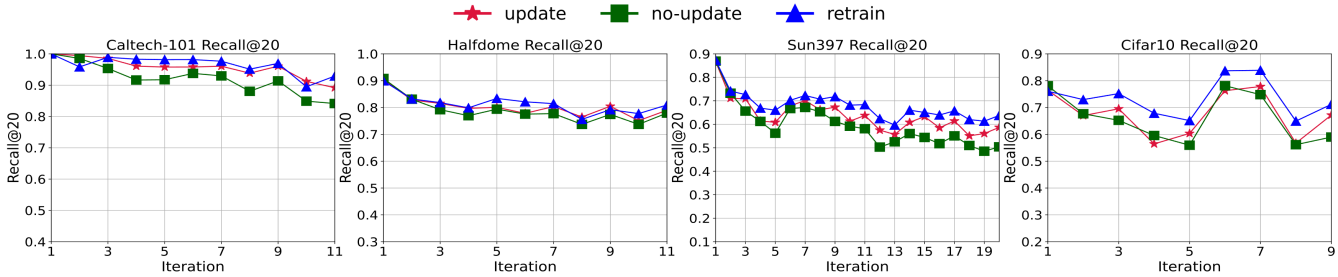

Figure 2: The effect of codebook update

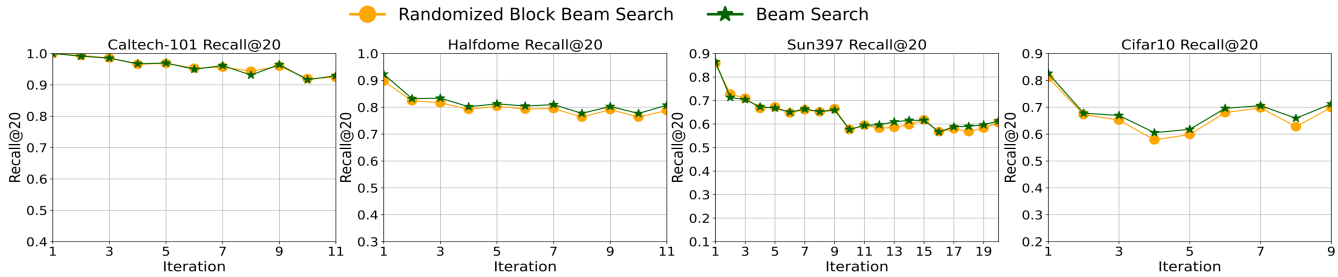

Figure 3: The effect of randomized block beam search on search accuracy

metric, and then update the codebooks. It is worth noting that, unlike online hash algorithms that need to recompute the hash codes after updating the parameters, our online algorithm only needs to encode the current new batch, and the previous data does not need to be recomputed. Streaming mode shares the same dataset preprocess setting with the batch mode.

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

6.3 Baselines

In this section, we introduce the baseline methods for comparison. Note that our algorithm does not use supervised information, we only compare online AQ with unsupervised online methods due to unfair comparison with supervised methods. Moreover, online PQ has been shown to greatly outperform online hashing [12, 13] and AdaptHash [5], so the comparison with them on the same datasets is not necessary anymore. According to our literature survey, unsupervised online hashing is mainly based on matrix sketching. Therefore, we compare our method with online PQ and sketching-based hashing.

- **Online Product Quantization [35]** , is the online version of product quantization and the state-of-the-art unsupervised learning to index methods. The number of codebooks is set to 8 and the number of codewords in each codebook is set to 256.
- **Online Sketching Hashing (OSH) [17]**, is the representative work of unsupervised online hashing, which maximizes the variance of every hashing bit among the sketched data. The code length is set to 64 bits, which occupies the same storage as quantization-based methods
- **Fast Online Sketching Hashing (Fast OSH) [8]**, improves the data sketching procedure in the OSH method with the fast subsampled randomized Hadamard transform. The code length is also set to 64 bits.

6.4 Comparison with Baselines

The comparison results with baselines are shown in Fig. 1. From this figure, we can make the following observations.

First, the proposed algorithm remarkably outperforms online product quantization. The main reason lies in the superiority of additive quantization to product quantization. Note that compared to product quantization, additive quantization does not decompose data space into orthogonal subspaces, and the codewords are of the same length as the input vectors and are generally not orthogonal to each other.

Second, both online AQ and online PQ outperforms sketch-based online hashing. This is because quantization-based methods reserve more information in original vectors than hashing-based methods. Concretely, quantization-based methods represent each data with a list of indexes with respect to codebooks, while hashing-based methods represent each data with a binary vector. Thus the representation capacity of quantization is much larger than hashing.

Finally, the search accuracy with online learned index usually decays when incrementally accommodating more and more data, but our method is much more robust than online PQ, OSH, and Fast OSH. This is based on the observation that our method remains more stable and even sometimes improves with more data incorporated for updating. This also indicates the effectiveness of the proposed approaches for updating codebooks.

6.5 The effectiveness of codebook update

To investigate the effectiveness of codebook update, we compare online AQ with two variants, one is to retrain the whole codebooks and the other one is not to update the codebooks at all. The results are shown in Fig. 2. This figure consistently shows that updating codebooks can work better than no update but worse than retrain as expected. However, the gap between update and retrain is not large. These results verify the effectiveness of the proposed approach for updating codebooks.

6.6 The effectiveness of randomized block beam search

Next, we study the effectiveness and efficiency of the proposed randomized block beam search. The results of efficiency on four datasets can be referred in Fig 4, where we vary the block size for randomize block beam search and see how the quantization error changes. From this figure, we see that randomized block beam search improves the efficiency of computing codes for the data. When the block size grows, randomized block beam search slows down while the quantization error is reduced. When the block size equals the number of codebooks, it is degenerated to beam search. We also report the search accuracy of online AQ on the four datasets Fig. 3, where beam search and randomized block beam search (F=5 and #iter=1) is used for computing codes for the data. This figure shows that randomized block beam search achieves a similar searching performance. Therefore, the proposed randomized block beam search strikes a better balance between efficiency and effectiveness for computing codes of the data.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose online Additive Quantization for accommodating incremental data, to support approximate nearest neighbor search in the dynamic database. To improve the efficiency of computing codes for each data, we develop randomized block beam search, which iteratively does beam search over a group of randomly selected codebooks. The empirical study demonstrates randomized block beam search is indeed faster than beam search at a small cost of quantization error and search accuracy. We derive theoretical results about regret bound, to guarantee the performance of online AQ. Finally, we evaluate the proposed algorithm with four real-world datasets, showing that online AQ remarkably outperforms the state-of-the-art online learning to index methods. In the future, we can investigate how to improve the efficiency of codebook update, with the techniques of such matrix sketching. Also, we are interested in studying how to perform online update for other indexes, such as graph and tree-based indexes.

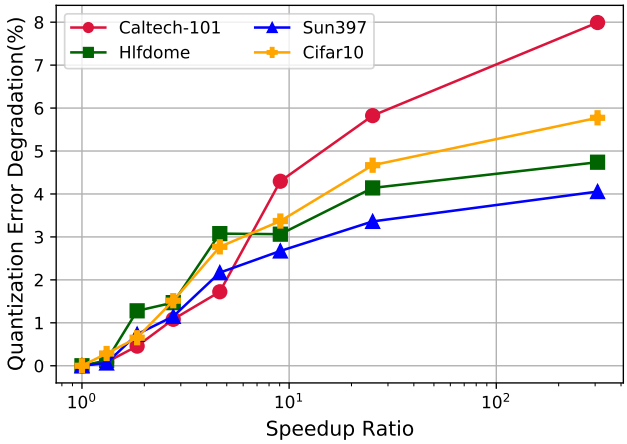

Figure 4: The effect of the varying block size in randomized block beam search.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *Proceedings of CVPR'14*. 931–938.
[2] Artem Babenko and Victor Lempitsky. 2015. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4240–4248.
[3] Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. 2017. Online supervised hashing. *Computer Vision and Image Understanding* 156 (2017), 162–173.
[4] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. 2017. Mihash: Online hashing with mutual information. In *Proceedings of the IEEE International Conference on Computer Vision*. 437–445.
[5] Fatih Cakir and Stan Sclaroff. 2015. Adaptive hashing for fast similarity search. In *Proceedings of the IEEE international conference on computer vision*. 1044–1052.
[6] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. 2004. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.* 33, 6 (2004), 1417–1440.
[7] Patrick Chen, Si Si, Sanjiv Kumar, Yang Li, and Cho-Jui Hsieh. 2018. Learning to Screen for Fast Softmax Inference on Large Vocabulary Neural Networks. In *International Conference on Learning Representations*.
[8] Xixian Chen, Irwin King, Michael R Lyu, et al. 2017. FROSH: FasteR Online Sketching Hashing.. In *UAI*.
[9] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 4 (2014), 744–755.

[10] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013).

[11] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*. PMLR, 3887–3896.

[12] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng. 2013. Online Hashing. In *IJCAI*. 1422–1428.

[13] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng. 2017. Online hashing. *IEEE transactions on neural networks and learning systems* 29, 6 (2017), 2309–2322.

[14] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

[15] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2012. Regularization techniques for learning with matrices. *The Journal of Machine Learning Research* 13, 1 (2012), 1865–1890.

[16] Weihao Kong and Wu-Jun Li. 2012. Isotropic hashing. In *Proceedings of NIPS'12*. 1646–1654.

[17] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu. 2015. Online sketching hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2503–2511.

[18] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[19] Edo Liberty. 2013. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 581–588.

[20] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems* 45 (2014), 61–68.

[21] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[22] Julieta Martinez, Joris Clement, Holger H Hoos, and James J Little. 2016. Revisiting additive quantization. In *European Conference on Computer Vision*. Springer, 137–153.

[23] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. 2018. A survey of product quantization. *ITE Transactions on Media Technology and Applications* 6, 1 (2018), 2–10.

[24] Stanislav Morozov and Artem Babenko. 2018. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems* 31 (2018), 4721–4730.

[25] Stephen Mussmann and Stefano Ermon. 2016. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*. PMLR, 2587–2596.

[26] Behnam Neyshabur and Nathan Srebro. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *Proceedings of ICML'15*. 1926–1934.

[27] Francesco Orabona, Koby Crammer, and Nicolo Cesa-Bianchi. 2015. A generalized online mirror descent with applications to classification and regression. *Machine Learning* 99, 3 (2015), 411–435.

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

[29] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[30] Hanan Samet. 2006. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann.

[31] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time Maximum Inner Product Search (MIPS). In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*. 2321–2329.

[32] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. 2015. Learning to hash for indexing big data—A survey. *Proc. IEEE* 104, 1 (2015), 34–57.

[33] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* (2017).

[34] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Proceedings of NIPS'09*. 1753–1760.

[35] Donna Xu, Ivor W Tsang, and Ying Zhang. 2018. Online product quantization. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2185–2198.

[36] Minjia Zhang, Xiaodong Liu, Wenhan Wang, Jianfeng Gao, and Yuxiong He. 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.

[37] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite Quantization for Approximate Nearest Neighbor Search. In *Proceedings of ICML'14*. 838–846.

[38] Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang. 2015. Sparse composite quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4548–4556.

Algorithm 3: Online Mirror Descent

Parameters: A sequence of strongly convex functions $f_0, f_1, \ldots,$ defined on a common convex domain $S \subseteq X$

Initialize: $\theta_1 \in X$;
for $t = 1, 2, \ldots$ **do**
  Choose $w_t = \nabla f_t^*(\theta_t)$;
  Observe $z_t \in X$;
  Update $\theta_{t+1} = \theta_t + z_t$;

A FENCHEL CONJUGATE

Given a closed and convex function f with domain S $\subseteq$ X, its Fenchel conjugate $f^* : \mathrm{X} \to \mathrm{R}$ is defined by $f^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in \mathrm{S}} \langle \boldsymbol{v}, \boldsymbol{u} \rangle - f(\boldsymbol{v})$. A generic norm of a vector $\boldsymbol{u} \in \mathrm{X}$ is denoted by $\|\boldsymbol{u}\|$. Its dual $\|\cdot\|_*$ is the norm defined by $\|\boldsymbol{v}\|_* = \sup_{\boldsymbol{u}} \{\langle \boldsymbol{u}, \boldsymbol{v} \rangle : \|\boldsymbol{u}\| \leq 1\}$. The Fenchel-Young inequality states that

$$f(\boldsymbol{u}) + f^*(\boldsymbol{v}) \geq \langle \boldsymbol{u}, \boldsymbol{v} \rangle \text{ for all } \boldsymbol{v}, \boldsymbol{u}.$$

A function f is β -strongly convex with respect to a norm $\|\cdot\|$ if for any u, v in its domain, and any $\boldsymbol{x} \in \partial f(\boldsymbol{u})$

$$f(\boldsymbol{v}) \geq f(\boldsymbol{u}) + \langle \boldsymbol{x}, \boldsymbol{v} - \boldsymbol{u} \rangle + \frac{\beta}{2} ||\boldsymbol{u} - \boldsymbol{v}||^2.$$

The Fenchel conjugate f^* of a β -strongly convex function f is everywhere differentiable and $\frac{1}{\beta}$ -strongly smooth. This means that, for all $\boldsymbol{u}, \boldsymbol{v} \in X$,

$$f^*(\boldsymbol{v}) \leq f^*(\boldsymbol{u}) + \left\langle \nabla f^*(\boldsymbol{u}), \boldsymbol{v} - \boldsymbol{u} \right\rangle + \frac{1}{2\beta} ||\boldsymbol{u} - \boldsymbol{v}||_*^2.$$

See also the paper [15] and references therein. A further property of strongly convex functions $f : S \to R$ is the following: for all $\boldsymbol{u} \in X$,

$$\nabla f^*(\boldsymbol{u}) = \mathrm{argmax}_{\boldsymbol{v} \in S} \langle \boldsymbol{v}, \boldsymbol{u} \rangle - f(\boldsymbol{v}).$$

This implies the useful identity.

$$f\left(\nabla f^*(\boldsymbol{u})\right) + f^*(\boldsymbol{u}) = \left\langle \nabla f^*(\boldsymbol{u}), \boldsymbol{u} \right\rangle.$$

Strong convexity and strong smoothness are key properties in the design of online learning algorithms.

B PROOF OF MAIN THEORETICAL RESULTS

B.1 Proof of Lemma 1

Proof. Let $\Delta_t = f_t^*(\boldsymbol{\theta}_{t+1}) - f_{t-1}^*(\boldsymbol{\theta}_t)$. Since the functions f_t^* are $\frac{1}{\beta_t}$ -strongly smooth with respect to $\|\cdot\|_{t,*}$, and recalling that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{z}_t$,

$$\begin{aligned}
\Delta_t &= f_t^*(\boldsymbol{\theta}_{t+1}) - f_t^*(\boldsymbol{\theta}_t) + f_t^*(\boldsymbol{\theta}_t) - f_{t-1}^*(\boldsymbol{\theta}_t) \\
&\leq f_t^*(\boldsymbol{\theta}_t) - f_{t-1}^*(\boldsymbol{\theta}_t) + \left\langle \nabla f_t^*(\boldsymbol{\theta}_t), \boldsymbol{z}_t \right\rangle + \frac{1}{2\beta_t} \|\boldsymbol{z}_t\|_{t,*}^2. \\
&= f_t^*(\boldsymbol{\theta}_t) - f_{t-1}^*(\boldsymbol{\theta}_t) + \langle \boldsymbol{w}_t, \boldsymbol{z}_t \rangle + \frac{1}{2\beta_t} \|\boldsymbol{z}_t\|_{t,*}^2
\end{aligned}$$

where we used the definition of $\boldsymbol{w}_t$ in the last step. On the other hand, we have

$$\sum_{t=1}^{T} \Delta_t = f_T^* (\boldsymbol{\theta}_{T+1}) - f_0^* (\boldsymbol{\theta}_1),$$

the Fenchel-Young inequality implies

$$f_T^* (\boldsymbol{\theta}_{T+1}) \geq \langle \boldsymbol{u}, \boldsymbol{\theta}_{T+1} \rangle - f_T(\boldsymbol{u}) = \sum_{t=1}^{T} \langle \boldsymbol{u}, z_t \rangle - f_T(\boldsymbol{u}).$$

Combining the upper and lower bound on Δ_t and summing over t we get

$$\sum_{t=1}^{T} \langle \boldsymbol{u}, z_t \rangle - f_T(\boldsymbol{u}) - f_0^* (\boldsymbol{\theta}_1) \leq \sum_{t=1}^{T} \{ f_t^* (\boldsymbol{\theta}_t) - f_{t-1}^* (\boldsymbol{\theta}_t) + < \boldsymbol{w}_t, z_t > + \frac{1}{2\beta_t} ||z_t||_{t,*}^2 \}$$

. We now prove the second statement. Recalling again the definition of $\boldsymbol{w}_t$, we have $f_t^* (\boldsymbol{\theta}_t) = \langle \boldsymbol{w}_t, \boldsymbol{\theta}_t \rangle - f_t (\boldsymbol{w}_t)$. On the other hand, the Fenchel-Young inequality implies that

$$-f_{t-1}^* (\boldsymbol{\theta}_t) \leq f_{t-1} (w_t) - \langle \boldsymbol{w}_t, \boldsymbol{\theta}_t \rangle .$$

Combining them, we get $f_t^* (\boldsymbol{\theta}_t) - f_{t-1}^* (\boldsymbol{\theta}_t) \leq f_{t-1} (\boldsymbol{w}_t) - f_t (\boldsymbol{w}_t)$, as desired. □

B.2 Lemma 2

Lemma 2. *Let $\boldsymbol{B}$ be an arbitrary full-rank matrix of size $n \times n$, let $\boldsymbol{x}$ an arbitrary vector, and let $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{x}\boldsymbol{x}^\top$. Then*

$$\boldsymbol{x}^\top \boldsymbol{A}^{-1} \boldsymbol{x} = 1 - \frac{det(\boldsymbol{B})}{det(\boldsymbol{A})}.$$

Proof. If $\boldsymbol{x} = (0, \ldots, 0)$, then the theorem holds trivially. Otherwise, we write

$$\boldsymbol{B} = \boldsymbol{A} - \boldsymbol{x}\boldsymbol{x}^\top = \boldsymbol{A} \left(\boldsymbol{I} - \boldsymbol{A}^{-1} \boldsymbol{x}\boldsymbol{x}^\top \right).$$

Hence, computing the determinant of the leftmost and rigthmost matrices,

$$\det(\boldsymbol{B}) = \det(\boldsymbol{A}) \det \left(\boldsymbol{I} - \boldsymbol{A}^{-1} \boldsymbol{x}\boldsymbol{x}^\top \right).$$

The right-hand side of this equation can be transformed as follows:

$$\det \left(\boldsymbol{I} - \boldsymbol{A}^{-1} \boldsymbol{x}\boldsymbol{x}^\top \right) = \det \left(\boldsymbol{A}^{\frac{1}{2}} \right) \det \left(\boldsymbol{I} - \boldsymbol{A}^{-1} \boldsymbol{x}\boldsymbol{x}^\top \right) \det \left(\boldsymbol{A}^{-\frac{1}{2}} \right) = \det \left(\boldsymbol{I} - \boldsymbol{A}^{-\frac{1}{2}} \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{A}^{-\frac{1}{2}} \right).$$

Hence, we are left to show that $\det \left(\boldsymbol{I} - \boldsymbol{A}^{-\frac{1}{2}} \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{A}^{-\frac{1}{2}} \right) = 1 - \boldsymbol{x}^\top \boldsymbol{A}^{-1} \boldsymbol{x}$. Letting $\boldsymbol{z} = \boldsymbol{A}^{-1/2} \boldsymbol{x}$, this can be rewritten as $\det \left(\boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^\top \right) = 1 - \boldsymbol{z}^\top \boldsymbol{z}$. It is easy to see that $\boldsymbol{z}$ is an eigenvector of $\boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^\top$ with eigenvalue $\lambda_1 = 1 - \boldsymbol{z}^\top \boldsymbol{z}$. Moreover, the remaining $d-1$ eigenvectors $u_2, \ldots, u_d$ of $\boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^\top$ form an orthogonal basis of the subspace of R^d orthogonal to $\boldsymbol{z}$, and the corresponding eigenvalues $\lambda_2 \ldots, \lambda_d$ are all equal to 1. Hence,

$$\det \left(\boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^\top \right) = \prod_{i}^{d} \lambda_i = 1 - \boldsymbol{z}^\top \boldsymbol{z},$$

which concludes the proof. □

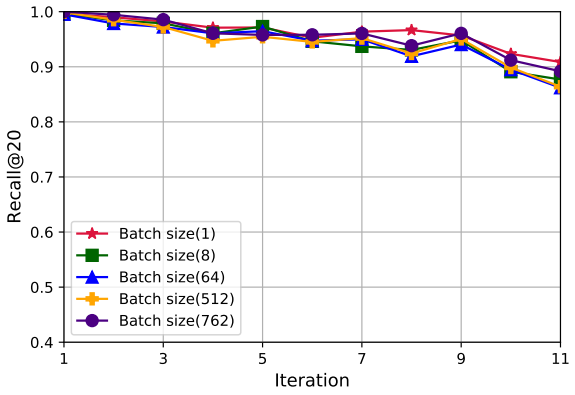

Figure 5: Comparison between the online version and the mini-batch version

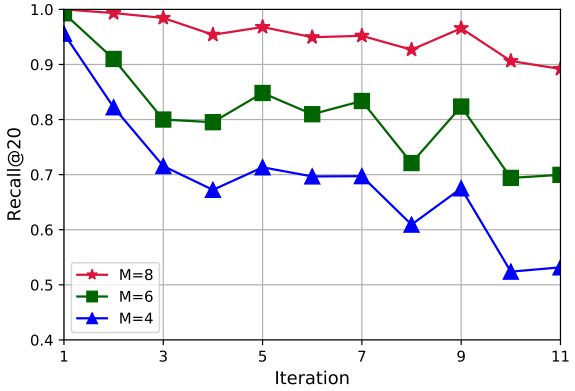

Figure 6: The effect of number of codebooks

C MORE EXPERIMENTS

In this section of the Appendix, we present more experimental results.

C.1 Online versus Mini-Batch

Our model can process streaming data either one at a time or in mini-batches. We compare these two versions of our model on the Caltech-101 dataset and show the results in Fig.5. When the batch size equals 1, only one data is processed each time. The number in the bracket is the number of data in the mini-batch. From this figure, we can see that the mini-batch version has similar search quality. It is reasonable to adopt the mini-batch version of online AQ for evaluation.

C.2 The effect of the number of codebooks

In this part, we investigate the effect of the number of codebooks on the searching performance and show the results in Fig.6. The figure shows that the larger number of codebooks leads to better

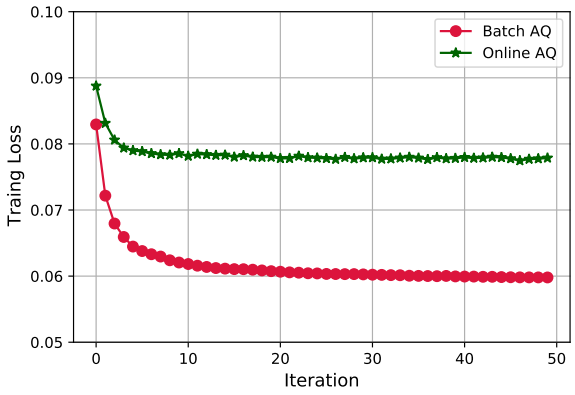

Figure 7: Convergence of online AQ

searching performance, which is quite consistent with our expectations.

C.3 Convergence Study

We also investigate the convergence of online AQ. The data in the whole dataset are input sequentially into online AQ. We run it for 50 effective iterations, which is defined as one pass through the whole dataset. In each iteration, AQ is also run within an iteration on the whole dataset. The quantization error of AQ and online AQ is plotted in Fig. 7 against the iterations. The figure shows that our model converges within tens of iterations, though the quantization error of online AQ is slightly larger than AQ. This indicates that codebooks learned from online AQ are similar to AQ, implying the effectiveness of the online AQ.