Self-Organizing Binary Encoding for Approximate Nearest Neighbor Search

Ezgi Can Ozan
Signal Processing Department
Tampere University of Technology
Tampere, Finland
ezgi.ozan@tut.fi

Serkan Kiranyaz
Electrical Engineering Department,
College of Engineering
Qatar University
Qatar
mkiranyaz@qu.edu.qa

Moncef Gabbouj
Signal Processing Department
Tampere University of Technology
Tampere, Finland
moncef.gabbouj@tut.fi

Xiaohua Hu
College of Computing and Informatics
Drexel University
Philadelphia, PA, USA
xh29@drexel.edu

***Abstract*—Approximate Nearest Neighbor (ANN) search for indexing and retrieval has become very popular with the recent growth of the databases in both size and dimension. In this paper, we propose a novel method for fast approximate distance calculation among the compressed samples. Inspiring from Kohonen's self-organizing maps, we propose a structured hierarchical quantization scheme in order to compress database samples in a more efficient way. Moreover, we introduce an error correction stage for encoding, which further improves the performance of the proposed method. The results on publicly available benchmark datasets demonstrate that the proposed method outperforms many well-known methods with comparable computational cost and storage space.**

Keywords- Approximate Nearest Neighbor Search; Quantization; Binary Feature Synthesis; Vector Compression, Self-Organizing Maps.

I. INTRODUCTION

The increase in the amount of daily generated data has become a significant research problem in recent years. As the size and dimension of the generated data grew, traditional methods began to fail to produce satisfactory results; therefore, new solutions specific for very large-scale datasets are desired. An important approach to handle the aforementioned problem is to develop approximate solutions. Approximate Nearest Neighbor Search (ANN) has become a noteworthy research topic in recent years for indexing and retrieval in very large-scale datasets [1]. ANN aims to compress the dataset samples as binary strings, then estimate the distance between the dataset samples and novel queries in a computationally efficient way.

Methods in the literature for ANN can be split into two main branches: Hashing and Vector Quantization (VQ). Hashing methods aim to approximate the distance between samples by calculating the Hamming distance between binary strings. Calculation of Hamming distance is very fast but since distances are integer values, the accuracy is inferior [2]–[6]. VQ based methods aim to approximate the distance between samples by using look-up tables of pre-calculated distances between learned codevectors and have proven to outperform Hashing based methods in terms of retrieval accuracy [7]–[11].

In this paper, we propose a novel VQ based binary encoding method, which performs ANN search on very large-scale datasets in a computationally efficient way. We inspire from Kohonen's self-organizing maps [12] and propose a hierarchical structure of several layers of quantization. We compare our method with well-known methods from the literature and our method outperforms those methods with comparable computational and storage costs.

The rest of the paper is organized as follows: In Section II, the problem formulation is defined and related works in the literature are introduced. In Section III, the proposed method is explained in detail. Section IV represents the findings of experiments on large scale datasets and finally in Section V the paper is concluded.

II. PROBLEM FORMULATION AND RELATED WORK

In this section, we first define VQ as an optimization problem. For a quantizer Q, given a set of D dimensional N input vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \in \mathbb{R}^{D \times N}$, the mean squared quantization error MSE_Q is defined as in (1).

$$MSE_Q = \frac{1}{N}\sum_{i}^{N} \|\boldsymbol{x}_i - Q(\boldsymbol{x}_i)\|^2 \quad (1)$$

Among the K codevectors in the codebook matrix $\boldsymbol{C} \in \mathbb{R}^{D \times K}$ of Q, the suitable one is selected by a binary selection vector $\boldsymbol{b}_i \in \{0,1\}^K$ is the binary selection vector, with $\|\boldsymbol{b}_i\| = 1$.

$$Q(\boldsymbol{x}_i) = \boldsymbol{C}\boldsymbol{b}_i \quad (2)$$

In ANN adaptations of VQ, the number of codevectors should be much greater than the traditional VQ. This is obtained by using several codebooks for quantization [1]. By using M different codebooks, the number of codevectors can be increased from K to K^M.

Product Quantization (PQ) by Jégou *et al.* [7] is among the first examples of VQ for ANN. The authors propose to divide the feature space into M subspaces, therefore, the optimization

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

is split into M problems and solved in each subspace separately. However, the division of the feature space requires the assumption of statistical independency among the subspaces, which is not realistic in practice. Several improvements on PQ have been proposed in order to overcome this drawback such as [8], [9]. The optimization problem can be formulated as given in (3).

$$MSE_Q^{(PQ_1)} = \min_{\{\boldsymbol{C}_1\},\{\boldsymbol{B}_1\}} \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{C}_1\boldsymbol{b}_{1_i}\|^2$$
$$\vdots \qquad (3)$$
$$MSE_Q^{(PQ_M)} = \min_{\{\boldsymbol{C}_M\},\{\boldsymbol{B}_M\}} \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{C}_M\boldsymbol{b}_{M_i}\|^2$$

A. Residual Vector Quantization

Residual Vector Quantization (RVQ) by Chen *et al.* [11] is another significant example for VQ for ANN, which follows a totally different approach than PQ. The authors propose to create hierarchical layers of quantization, where in each layer the residuals of the previous layers are quantized. So the optimization problem is again split into M problems and solved separately. RVQ can be formulated as given in (4).

$$MSE_Q^{(RVQ_1)} = \min_{\{\boldsymbol{C}_1\},\{\boldsymbol{B}_1\}} \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\boldsymbol{x}_i - \sum_{m=1}^{1}\boldsymbol{C}_m\boldsymbol{b}_{m_i}\right)\right\|^2$$
$$\vdots \qquad (4)$$
$$MSE_Q^{(RVQ_M)} = \min_{\{\boldsymbol{C}_M\},\{\boldsymbol{B}_M\}} \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\boldsymbol{x}_i - \sum_{m=1}^{M}\boldsymbol{C}_m\boldsymbol{b}_{m_i}\right)\right\|^2$$

As it can be seen in the equation above, even the problem is split into M layers, the optimization in each layer depends on the solutions of the previous layers. In other words, the selection of codevectors in each layer highly affects the selection of the codevectors in the next layers.

In RVQ, each problem is solved using the K-Means algorithm. K-Means algorithm converges to a local minima in each layer, but when upper layers overfit to the data, the contribution of lower layers decreases, leading to inferior codebooks. In this paper, we introduce a structure for each codebook, to prevent overfitting in higher layers and obtain better overall quantization.

III. The Proposed Method

The proposed method aims to improve the training of codebooks by imposing a structure which is inspired from the self-organizing maps in order to prevent overfitting.

A. Self-Organizing Maps

Self-organizing maps (SOM) can be considered as neural networks, which spatially order the responses of neurons. It is inspired from the biological fact that, in brain, the cells which are together generally respond together. In this iterative algorithm, the neurons are connected to each other by a predefined topology. When one neuron is updated, the neurons within the predefined neighborhood are updated as well as a result of the imposed structure.

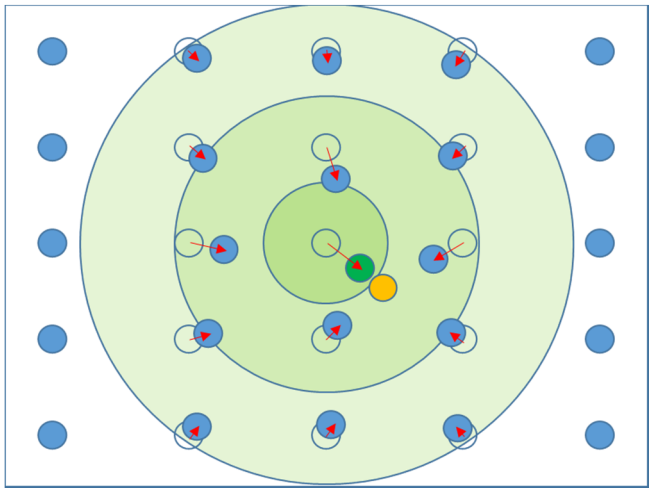

Figure 1. Illustration of SOM neuron update.

Training of SOM is a competitive learning process, i.e., for each input vector, the best matching neuron is updated to match even more closely to the corresponding vector. For an input vector $\boldsymbol{x}$, the best matching neuron is defined as given in (5).

$$\mathrm{k}^* = \underset{k}{\operatorname{argmin}}(\|\boldsymbol{x} - \boldsymbol{c}_k\|^2) \qquad (5)$$

As no closed form solution is available for the problem above, iterative approximations are used. Using the stochastic gradient descent approach, the update equation can be formulated as given in (6).

$$\boldsymbol{c}_k(t+1) = \boldsymbol{c}_k(t) - \gamma(t)\nabla_{\boldsymbol{c}_k}(\|\boldsymbol{x} - \boldsymbol{c}_k\|^2) \qquad (6)$$

where $\nabla_{\boldsymbol{c}_k}$ is the gradient operation and $\gamma(t)$ is the learning rate. The corresponding gradient can be formulated as given in (7).

$$\nabla_{\boldsymbol{c}_k}(\|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2) = 2(\boldsymbol{c}_k - \boldsymbol{x}) \qquad (7)$$

Using (5), (6) and (7), the update equation for the self-organizing maps with the imposed topology can be formulated as given in (8), where $\boldsymbol{c}_{k^*}$ is the winner neuron and $\mathcal{N}_{\boldsymbol{c}_{k^*}}$ represents the set of neighboring neurons, as defined by the topology [12].

$$\boldsymbol{c}_k(t+1) = \begin{cases} \boldsymbol{c}_k(t) - \gamma(t)(\boldsymbol{c}_k - \boldsymbol{x}) \\ \boldsymbol{c}_k(t) \end{cases} \begin{matrix} if\ k \in \mathcal{N}_{\boldsymbol{c}_{k^*}} \\ else \end{matrix} \qquad (8)$$

The neighborhood can be defined by a Gaussian kernel function as given in (9) and (10), where $\mathrm{h}_{(k,k^*)}(t)$ is the kernel function, r_k is the position vector of the k^{th} neuron, $h_0(t)$ and $\sigma(t)$ are two suitable functions which decrease with time.

$$\boldsymbol{c}_k(t+1) = \boldsymbol{c}_k(t) - h_{(k,k^*)}(t)(\boldsymbol{c}_k - \boldsymbol{x}) \qquad (9)$$

$$h_{(k,k^*)}(t) = h_0(t)e^{-\|r_k - r_{k^*}\|^2/\sigma(t)^2} \qquad (10)$$

Alternatively, the distance between neurons can be used to define the neighborhood relations [12]. In our approach we follow this method in order to discard the parameter $\sigma(t)$, which is dataset dependent. We define a number of nearest neurons to

This work is funded by the TEKES project "Big Data Analysis" 32.20.40.2/15 and Tieto Finland Oy.

TABLE I. PSEUDO-CODE FOR MULTI-DIMENSIONAL TOPOLOGY

Given: $\boldsymbol{X}$: the set of input samples K: total number of neurons
k_l: The number of centroids assigned to dimension l. $\boldsymbol{V}$: vector of standard deviations, V_l: The standard deviation for dimension l. • Apply PCA on $\boldsymbol{X}$ and obtain $\boldsymbol{V}$ • while $\prod 2^{k_l} < K$, ○ $l^* = \max_{\boldsymbol{l}} V_{\boldsymbol{l}}$, ○ $V_{l^*} = V_{l^*}/2$ ○ $k_{l^*} = k_{l^*} + 1$ • Obtain 2^{k_l} centroids on each dimension l • Concatenate all permutations of obtained centroids to initialize the positions of neurons.

the winner neuron as the neighbor set and decrease this number exponentially with the iterations.

The neuron update process for SOM is illustrated in Figure 1. As it can be seen, the winner neuron (green disc) and all the neighboring neurons around it (blue discs) are updated to match even better (to move closer) to the given sample (yellow disc).

B. *Multi-Dimensional Topology*

Generally self-organizing maps define the network topology in a 2D form [13]. Here in this paper, we propose a multidimensional initialization for the imposed structure based on the Principal Component Analysis (PCA). As the number of neurons are predefined in SOM, we aim to distribute these neurons uniformly within the feature space.

The proposed algorithm starts with transforming the training data using PCA, in order to obtain the principal components and the corresponding standard deviations. On each principal component, we perform 1D clustering using Max-Lloyd quantization, which is very similar to K-Means [14]. To achieve the uniform distribution the number of centroids for each dimension is kept directly proportional to the standard deviation. After the centroids are distributed uniformly in the transform space, concatenating the centroids of each dimension for all the permutations, initial positions of the neurons are obtained.

Here note that, the number of all the permutations is, $\lambda_1 \times \lambda_2 \times \ldots \times \lambda_L$, where λ_l is the number of centroids on the l^{th} dimension and L is the number of centroid appointed dimensions. In order to ensure that the total number of neurons $K = \prod_1^L \lambda_l$, we select K to be a power of 2, and we also distribute the centroids to the dimensions with the powers of 2. The algorithm that is used for the distribution of neurons among the feature space is given in TABLE I.

In RVQ's top down hierarchical scheme [11], each codevector in a lower layer is a residual of the corresponding upper layer. In other words, we can consider the lower layer codevectors as replicated and put around each and every one of the higher layer codevectors. This is illustrated in Figure 2. As a result of this scheme, the positions of lower layer codevectors

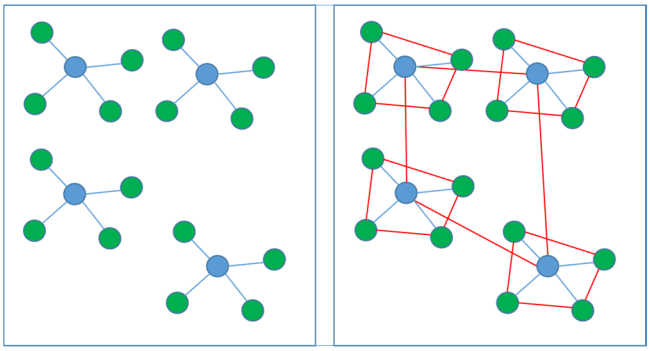

Figure 2. RVQ connections (left) vs SOBE connections (right).

are highly dependent on the positions of higher layer codevectors in the feature space. While this method has proved to be a very efficient way of exponentially increasing the total number of codevectors, the connection between the higher and lower layers also helps the distribution of codevectors among the space [11].

However, in the proposed method, we aim to add another connection among the codevectors which are in the same layer, by using SOMs. It is an intuitive expectation that, the connections between the codevectors of the same and different layers will improve the quality of training, leading to lower quantization error and better retrieval results.

C. *Self-Organized Binary Embedding (**SOBE**)*

As mentioned earlier, the proposed method uses SOMs in a top-down hierarchical order to perform quantization on vectors. At each layer, codevectors are calculated by training a SOM and then the best matching codevector (neuron weight) is subtracted from each sample in order to obtain the residuals similar to [11]. The residuals are carried to the next layer of quantization. Again for this layer, a SOM is trained and the best matching codevector of the new SOM is subtracted from the residuals in order to obtain the next set of residuals and so on. This is repeated until the final layer is reached.

Each layer of SOM is trained on the residuals of the previous layer, hence decreases the quantization error. Thanks to the initial structure of SOM, overfitting is evaded at each layer. Preventing the overfitting on upper layers, we can obtain a better quantization scheme, which utilizes the lower layers more efficiently.

D. *Encoding Correction*

Since our method uses the same hierarchical structure with RVQ, it is possible to propose a similar encoding method. The hierarchical encoding in RVQ takes place as follows: First the winner codevector for a layer is obtained, then the corresponding residual is calculated and passed on to the next layer. This is repeated for each layer.

In order to further improve the quality of encoding, we propose an additional encoding correction step on top of the hierarchical encoding approach mentioned above. For each layer, all the winner codevectors except the codevector of a given layer are subtracted from the given sample and the winner codevector for the given layer is recalculated. If the distance

TABLE II. RESULTS FOR SIFT1M, 32-BIT CODES

	recall@1	**recall@10**	**recall@100**
PQ	0.052	0.230	0.595
CKM	0.068	0.273	0.658
OCKM	NA	**0.348**	**0.742**
RVQ	NA	NA	NA
SOBE	**0.010**	**0.348**	0.731

TABLE III. RESULTS FOR SIFT1M, 64-BIT CODES

	recall@1	**recall@10**	**recall@100**
PQ	0.224	0.599	0.924
CKM	0.243	0.638	0.940
OCKM	0.273	0.680	0.945
RVQ	0.257	0.653	0.946
SOBE	**0.282**	**0.701**	**0.962**

TABLE IV. RESULTS FOR GIST1M, 32-BIT CODES

	recall@1	**recall@10**	**recall@100**
PQ	0.023	0.068	0.176
CKM	0.054	0.142	0.396
OCKM	NA	0.172	**0.468**
RVQ	NA	NA	NA
SOBE	**0.064**	**0.189**	0.403

TABLE V. RESULTS FOR GIST1M, 64-BIT CODES

	recall@1	**recall@10**	**recall@100**
PQ	0.076	0.218	0.504
CKM	0.118	0.334	0.715
OCKM	0.130	0.358	**0.719**
RVQ	0.113	0.325	0.676
SOBE	**0.136**	**0.360**	0.705

between the given sample and the corrected encoding is less than the initial encoding, then the corrected one is proposed as the final encoding.

IV. EXPERIMENTS

We test the performance of the proposed method in two publicly available datasets, SIFT1M and GIST1M [7]. Both datasets consist of 1 Million samples. SIFT1M consists of 128-dimensional SIFT vectors and GIST1M consists of 960-dimensional GIST vectors. We use the given training sets for training the codebooks and perform exhaustive search with the given queries. We use the parameters of $K = 256$, $M = 8$ for 64-bits and $M = 4$ for 32-bits coding.

A. ANN Search

The performance of the proposed method, **SOBE**, is compared against *Product Quantization* (**PQ**) [7], *Cartesian K-Means* (**CKM**) [9], *Optimized Cartesian K-Means* (**OCKM**) [10] and *Residual Vector Quantization* (**RVQ**) [11]. We also use **recall@R** performance metric, which is the average recall for the nearest neighbor in the first **r** retrievals, similar to the other methods in literature [1]. The results of the competing methods are taken from the provided figures in the original publications. The 32-bit and 64-bit encoding performances for SIFT1M dataset are given in TABLE II. and in TABLE III. . For GIST1M, the results are presented in TABLE IV. and in TABLE V. respectively.

As it can be seen in the results from the corresponding tables, the proposed method outperforms the compared methods for **recall@1** and **recall@10** in all tests. For **recall@100** the proposed method is outperformed by **OCKM** except SIFT1M 64-Bit test. The tests prove that the proposed enhancements on training and encoding have improved the nearest neighbor search performance.

In TABLE VI. the effect of encoding correction improvement has been shown in comparison with standard encoding on Sift1M dataset using 64-bit codes. As it can be seen, the proposed encoding method significantly improves the performance.

TABLE VI. EFFECT OF ENCODING CORRECTION ON SIFT1M, 64-BIT CODES

	recall@1	**recall@10**	**recall@100**
Standard	0.275	0.691	0.961
Corrected	**0.282**	**0.701**	**0.962**

B. Computational and Storage Costs

The computational cost of encoding of the proposed algorithm can be analyzed as follows: There are M hierarchical layers and for each layer, a given sample is compared with K codevectors of D dimensions each. The correction stage costs almost the same as encoding stage, so in total the cost of encoding is, $O(2MKD)$.

As it can be seen in TABLE VII. , the computational cost of encoding of the proposed method is higher than RVQ, PQ and CKM and comparable to OCKM, whereas the proposed method performs significantly better. In terms of storage costs, our method has the same requirements with RVQ, which is comparable to the storage requirements of the compared methods. The detailed comparison of storage costs are represented in TABLE VIII.

TABLE VII. COMPUTATIONAL COSTS FOR ENCODING

PQ	$O(KD)$
CKM	$O(D^2 + KD)$
OCKM	$O(10KD)$
RVQ	$O(MKD)$
SOBE	$O(2MKD)$

TABLE VIII. STORAGE COSTS

PQ	$O(KD)$
CKM	$O(D^2 + KD)$
OCKM	$O(D^2 + 2KD)$
RVQ	$O(MKD)$
SOBE	$O(MKD)$

V. CONCLUSIONS

In this paper, we propose a novel hierarchical vector quantization method, which splits the quantization problem into hierarchical layers and trains a SOM at each layer. The obtained residuals are quantized in the next layer, providing reduction in quantization error and efficient encoding. Moreover, we propose a code correction step in order to decrease the quantization error further. The experimental results show that our method performs better than the compared methods with comparable encoding and storage costs. As a future work we aim to investigate the performance of our method on billion scale datasets, and further test our algorithm on class labeled datasets using k-NN classification.

REFERENCES

[1] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for Similarity Search: A Survey," in *arXiv preprint*, 2014, p. :1408.2927.

[2] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*, 2010, pp. 3424–3431.

[3] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *CVPR*, 2011, pp. 817–824.

[4] P. Li, M. Wang, and J. Cheng, "Spectral hashing with semantically consistent graph for image indexing," *IEEE Trans. Multimed.*, vol. 15, no. 1, pp. 141–152, 2013.

[5] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recognit. Lett.*, vol. 31, no. 11, pp. 1348–1358, Aug. 2010.

[6] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 33–47, Jan. 2014.

[7] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–28, Jan. 2011.

[8] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized Product Quantization.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, pp. 1–12, Dec. 2014.

[9] M. Norouzi and D. J. Fleet, "Cartesian K-Means," in *CVPR*, 2013, pp. 3017–3024.

[10] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen, and S. Li, "Optimized Cartesian K-Means," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 180–192, Jan. 2015.

[11] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11259–11273, 2010.

[12] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

[13] M. Attik and L. Bougrain, "Self-organizing Map Initialization," in *ICANN*, 2005, pp. 357–362.

[14] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.