Discriminative Quantization for Fast Image Search

Sepehr Eghbali Ladan Tahvildari
University of Waterloo
{sepehr.eghbali, ladan.tahvildari}@uwaterloo.ca

Abstract

Recent decade has witnessed a growing surge of research on encoding high-dimensional objects with compact discrete codes. In this paper, we present a new supervised quantization technique to learn discriminative and compact codes for large scale retrieval tasks. To achieve fast and accurate search, the proposed algorithm learns a discriminative embedding of the input points and at the same time encodes the embedded points with compact codes to reduce storage cost.

1. Introduction

Recently, compact coding techniques have become increasingly popular due to their ability to compress high dimensional data using discrete and compact representation. With this view, unsupervised compact coding techniques try to find codes that are the well aligned with the Euclidean distances of the input vectors. Literature abounds with examples of unsupervised binary hashing and quantization (see [13] for an overview). Parallelly, supervised compact coding, specially supervised binary hashing, has been the topic of much work in recent years [3]. In the supervised setting, the goal is to encode input vectors with compact codes which are faithful to a notation of semantic similarity.

One shortcoming of binary hashing techniques is that there are only a few possible distances between binary codes which results in limited ability to describe the distances between data points. On the other hand, quantization-based techniques can produce extremely large number of possible distances by decomposing the space into a Cartesian product of subspaces and quantizing each subspace separately [4, 11, 8]. However, this comes at the cost of slightly higher query time in quantization techniques. While unsupervised quantization has been extensively studied, only a few recent studies have addressed its supervised counterpart [14].

In this paper, we put forward a new supervised quantization technique, called *Discriminative Quantization (DQ)*, that incorporates a form of a triplet loss function to address the shortcomings of existing techniques. Our approach performs two simultaneous tasks in order to achieve fast and scalable nearest neighbor search. First task is learning a mapping from the input space to a discriminative Euclidean space where distances directly correspond to the semantic similarity. The mapping is also responsible for enhancing the quantilzibilty of input vectors. The second task is quantizing the mapped vectors in order to achieve fast distance computation.

2. Formulation

Given a query $\mathbf{q} \in \mathbb{R}^d$ and a set of n points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$ with each point associated with a class label $y_i \in \{1, \ldots, c\}$, the goal is to preprocess the data in order to find the K nearest items to $\mathbf{q}$ as fast as possible such that the found items share the same label with the query. In this paper, we address this problem in its approximate setting in which $\mathbf{x}_i$s are approximated with discrete compact codes and nearest neighbor search is performed among such codes.

2.1. Semantic Discrimination

We strive for an embedding function $f(\mathbf{x}; \mathbf{w}) : \mathbb{R}^d \rightarrow \mathbb{R}^p$ parameterized with vector $\mathbf{w}$, such that the squared distance between the mapped points with the same class label is smaller than the squared distance between a pair of data points with different labels. In our approach, the parameters of f are trained on the triplet data $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) \in \mathcal{T}$ such that $\mathbf{x}$ and $\mathbf{x}^+$ are from the same class label while $\mathbf{x}^-$ is from a different class. Formally, we want to:

$$\|f(\mathbf{x}; \mathbf{w}) - f(\mathbf{x}^+; \mathbf{w})\|_2^2 + \alpha < \|f(\mathbf{x}; \mathbf{w}) - f(\mathbf{x}^-; \mathbf{w})\|_2^2 \quad \forall (\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) \in \mathcal{T}, \tag{1}$$

where α is a margin that in enforced between positive and negative pairs and $\mathcal{T}$ is the set of triplets. Therefore, the loss function takes the form of:

$$L(\mathbf{w}) = \sum_{(\mathbf{x},\mathbf{x}^+,\mathbf{x}^-)\in\mathcal{T}} [\|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x}^+;\mathbf{w})\|_2^2 - \|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x}^-;\mathbf{w})\|_2^2 + \alpha]_+, \tag{2}$$

where $[.]_+ = \max(0,.)$ is the standard hinge loss function.

2.2. Fast Distance Computation

To achieve fast distance computation, we propose to employ the state-of-the-art composite quantization (CQ) [16], a member of multi-codebook quantization (MCQ) family. The idea behind MCQ is to create multiple dictionaries (say m of them) and learn h codewords per dictionary with the goal of minimizing the reconstruction error. Formally, the product quantization based techniques aim to minimize the quantization error which in our problem takes the following form:

$$Q(\mathbf{C},\mathbf{B}) = \sum_{i=1}^{n} \|f(\mathbf{x}_i;\mathbf{w}) - \sum_{j=1}^{m} \mathbf{C}_j \mathbf{b}_{ij}\|_2^2 \quad \text{s.t.} \quad \mathbf{b}_{ij} \in \{0,1\}^h, \|\mathbf{b}_{ij}\|_1 = 1, \tag{3}$$

where $\mathbf{b}_{ij}$ is a binary vector that selects 1-of-h encodings from the j-th dictionary, $\mathbf{C}_j \in \mathbb{R}^{p\times h}$ is a matrix whose columns represent the codewords of the j-th dictionary. Also, $\mathbf{C} = [\mathbf{C_1},\ldots,\mathbf{C}_m] \in \mathbb{R}^{p\times mh}$ denotes the matrix of dictionaries and $\mathbf{B} = [\mathbf{b}_1,\ldots,\mathbf{b}_n] \in \{0,1\}^{h\times nm}$ denotes the matrix of indices where $\mathbf{b}_i = [\mathbf{b}_{i1}^T,\ldots,\mathbf{b}_{im}^T]^T$.

Optimizing (3) in general is intractable. Therefore, different techniques introduce various constraints to make the optimization feasible. In particular, CQ introduces an extra constraint such that for each $\mathbf{x}_i$ we have $\sum_{j\neq t} \mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it} = \epsilon$ known as the *constant inter-dictionary-element-product* constraint.

2.3. Optimization Problem

In our study, we perform simultaneous representation learning and quantization by integrating the introduced triplet and quantization losses. By adding simple regularizer on the vector of unknown parameter $\mathbf{w}$, the overall objective function is given as follows:

$$\min_{\mathbf{w},\mathbf{C},\{\mathbf{b}_i\}_{i=1}^n,\epsilon} L(\mathbf{w}) + \lambda Q(\mathbf{C},\mathbf{B}) + \frac{\gamma}{2}\|\mathbf{w}\|_2^2 \quad \text{s.t.} \sum_{j\neq t} \mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it} = \epsilon \quad \forall i \in [n], \tag{4}$$

where λ and γ are trade-off parameters between the triplet loss, quantization loss and the regularizer.

Intuitively, the transformation f is responsible for improving both the semantic discrimination and quantizability of the transformed vectors. Note that not all vectors can be quantized efficiently with vector quantization. If input vectors do not exhibit a cluster structure then they cannot be quantized accurately [1].

2.4. Querying

During the search phase, given the query $\mathbf{q}$, first the transformation f is applied to query vector $\mathbf{q}$, $\mathbf{q}' = f(\mathbf{q};\mathbf{w})$, and then the distance between $\mathbf{q}'$ and transformed items, $\mathcal{X}' = \{\mathbf{x}_1',\ldots,\mathbf{x}_n'\}$ where $\mathbf{x}_i' = f(\mathbf{x}_i;\mathbf{w})$, are approximated using the codewords that they belong to:

$$\|\mathbf{q}' - \mathbf{x}_i'\|_2^2 \approx \|\mathbf{q}' - \mathbf{C}\mathbf{b}_i\|_2^2 = \sum_{j=1}^{m} \|\mathbf{q}' - \mathbf{C}_j \mathbf{b}_{ij}\|_2^2 - (m-1)\|\mathbf{q}'\|_2^2 + \sum_{j\neq t}^{m} \mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it}. \tag{5}$$

For a given query, the term $(m-1)\|\mathbf{q}'\|_2^2$ is constant for all $\mathbf{x}_i'$s. Also, because of the introduced constant inter-dictionary-element-product condition the third term, $\sum_{j\neq t} \mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it}$, is also constant (equal to ϵ). Therefore, finding the nearest neighbors to $\mathbf{q}$ can be achieved by only computing the distance between $\mathbf{q}'$ and the selected dictionary elements. The main advantage of product quantization based techniques is that they enable fast distance computation by precomputing the distances between $\mathbf{q}'$ and all dictionary elements and saving them in an $m \times h$ lookup table at the beginning of the query phase. Doing this, computing $\sum_{j=1}^{m} \|\mathbf{q} - \mathbf{C}_j \mathbf{b}_{ij}\|_2^2$ boils down to m distance lookups and m addition operations.

2.5. Optimization Procedure

The optimization consists of 4 groups of unknown variables: parameters of the transformation function $\mathbf{w}$, dictionaries $\mathbf{C}$, binary indicator vectors $\mathbf{B}$ and the constant ϵ. Following [16], we combine the objective function with the constraint using the quadratic penalty method:

$$\min_{\mathbf{w},\mathbf{C},\{\mathbf{b}_i\}_{i=1}^n,\epsilon} J = L(\mathbf{w}) + \lambda Q(\mathbf{C},\mathbf{B}) + \frac{\gamma}{2}\|\mathbf{w}\|_2^2 + \mu \sum_{i=1}^{n} \left(\sum_{j\neq t}^{m} (\mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it} - \epsilon)^2 \right). \tag{6}$$

To optimize the parameters, we use alternative optimization in which each iteration alternatively updates $\mathbf{w}, \mathbf{B}, \mathbf{C}$ and ϵ. The details are given below:

Update B. Fixing $\mathbf{w}, \mathbf{C}$ and ϵ, it is easy to see that the choice of indicator vector for the ith data point, $\mathbf{b}_i$, is independent of all other points, $\{\mathbf{b}_t\}_{t\neq i}$. As a result, the optimization problem can be decomposed into n independent subproblems.

$$\psi(\mathbf{b}_i) = \|f(\mathbf{x}_i;\mathbf{w}) - \mathbf{C}\mathbf{b}_i\|_2^2 + \mu \sum_{j\neq t}^{m} (\mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it} - \epsilon)^2. \tag{7}$$

In general, this optimization problem is NP-hard. Here we again use alternative optimization and solve each of the j-th subvectors $\{\mathbf{b}_{ij}\}_{j=1}^{h}$ alternatively. For updating $\mathbf{b}_{ij}$, we fix $\{\mathbf{b}_{it}\}_{t\neq j}$ and then exhaustively check all codewords in $\mathbf{C}_j$ (meaning that we try all possible values of $\mathbf{b}_{ij}$) and select the one that minimizes the objective function in (7).

Update ϵ. It is easy to see that the objective function in (6) is quadratic in ϵ and we have a closed-form solution to ϵ:

$$\epsilon = \frac{1}{n} \sum_{i=1}^{n} \sum_{j\neq t}^{h} \mathbf{b}_{ij}^T \mathbf{C}_j^T \mathbf{C}_t \mathbf{b}_{it}. \tag{8}$$

Update $\mathbf{C}$. Fixing $\mathbf{w}, \mathbf{B}$ and ϵ, the optimization problem is an unconstraint nonlienar optimization problem with respect to $\mathbf{C}$. To solve it, we use L-BFGS algorithm which is the limited version of Fletcher-Goldfarb-Shanno (BFGS) algorithm and has a publicly available implementation[1]. L-BFGS takes the objective function and its derivative with respect to the variable of interest and iteratively updates the variable. The partial derivative of (6) with respect to $\mathbf{C}_t$ is:

$$\begin{aligned} &\sum_{i=1}^{n} \Big[-2 \sum_{j=1}^{h} (\mathbf{x}_i - \mathbf{C}_j \mathbf{b}_{ij}) \mathbf{b}_{it}^T + \\ &4\mu \sum_{k\neq j}^{m} (\mathbf{b}_{ik}^T \mathbf{C}_k^T \mathbf{C}_j \mathbf{b}_{ij} - \epsilon)(\sum_{j=1, j\neq t}^{m} \mathbf{C}_j \mathbf{b}_{ij}) \mathbf{b}_{it}^T \Big]. \end{aligned} \tag{9}$$

Update $\mathbf{w}$: To learn $\mathbf{w}$, we use stochastic gradient descent. The gradient of the objective function J with respect to $\mathbf{w}$ can be computed as follows. Let $s((\mathbf{x}, \mathbf{x}^+, \mathbf{x}); \mathbf{w}) = \|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x}^+;\mathbf{w})\|_2^2 - \|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x}^-;\mathbf{w})\|_2^2 + \alpha$, and let I_i denote the i-th triplet in τ then we have:

$$\frac{\partial J}{\partial \mathbf{w}} = \sum_{i=1}^{|\tau|} g(I_i, \mathbf{w}) + \lambda \sum_{i=1}^{n} 2(f(\mathbf{x}_i;\mathbf{w}) - \mathbf{C}\mathbf{b}_i) + \gamma \mathbf{w} \tag{10}$$

$$g(I_i, \mathbf{w}) = \begin{cases} \frac{\partial s(I_i, \mathbf{w})}{\partial \mathbf{w}} & s(I_i, \mathbf{w}) > 0 \\ 0 & s(I_i, \mathbf{w}) \leq 0 \end{cases}$$

$$\begin{aligned} \frac{\partial s}{\partial \mathbf{w}} =& 2(f(\mathbf{x}) - f(\mathbf{x}^+)) \frac{\partial f(\mathbf{x}) - \partial f(\mathbf{x}^+)}{\partial \mathbf{w}} - \\ & 2(f(\mathbf{x}) - f(\mathbf{x}^-)) \frac{\partial f(\mathbf{x}) - \partial f(\mathbf{x}^-)}{\partial \mathbf{w}}. \end{aligned} \tag{11}$$

[1]users.iems.northwestern.edu/~nocedal/lbfgsb.html

From the above derivations, it is clear that the gradient on each input triplet can be easily computed given the values of $f(\mathbf{x};\mathbf{w}), f(\mathbf{x}^+;\mathbf{w}), f(\mathbf{x}^-;\mathbf{w})$ and $\frac{\partial f(\mathbf{x};\mathbf{w})}{\partial \mathbf{w}}, \frac{\partial f(\mathbf{x}^+;\mathbf{w})}{\partial \mathbf{w}}, \frac{\partial f(\mathbf{x}^-;\mathbf{w})}{\partial \mathbf{w}}$.

2.6. Triplet Selection.

For large-scale datasets, it is impossible to train the algorithm on all $O(n^3)$ possible triplets. Moreover, generating all triplets would result in many triplets that are easily satisfied (*i.e.* fulfilling the constraint in (1)). Such triplets do not contribute to the training procedure and result in slower optimization as they would still be passed through the transformation f. Therefore, it is crucial to choose hard triplets that are active and contribute to improving the model.

To form the triplets, in our experiments we use mini-batches of 200 datapoints. For each mini-batch, we form 200 triplets by randomly selecting pairs $(\mathbf{x}, \mathbf{x}^+)$ such that both items have the same class label. Then, for each pair, we sort all the transformed vectors in the batch based on their Euclidean distance from $f(\mathbf{x})$ in ascending order. Finally, we find $\mathbf{x}^+$ in the ordering and select the first item after it that has a different class label from $\mathbf{x}$, as $\mathbf{x}^-$. Although it may seem more intuitive to select the hardest negative (*i.e.* is the last item in the ordering with a different class label compared to $\mathbf{x}$), we observed that it can make the optimization algorithm stuck in bad local minima early on in training.

3. Experiments

To gauge the performance of the proposed supervised quantization technique, we conduct experiments on two widely used large-scale face image datasets. We compare our technique with several state-of-the-art supervised hashing and quantization methods.

3.1. Experimental Setting

Our method, denoted by DQ, is compared with six state-of-the-art supervised compact coding methods: supervised discrete hashing (SDH) [12], FastHash [6], binary reconstructive embedding (BRE) [5], minimal loss hashing (MLH) [10], supervised quantization (SQ) [14], iterative quantization (ITQ) [2]. For all algorithms, we use their public implementation expect that for SQ we implemented it in Python as its public implementation is not available.

We use a two-layer neural network with 500-dimensional hidden layer with weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ such that the transformation function can be expressed as $f(\mathbf{x}) = \tanh(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{x}))$. We also adopt the kernel based representation [12] for SDH and SQ as it has been shown to boost the performance.

Parameter Settings. The objective function in (6) has three trade-off parameters: λ for the quantization loss term,

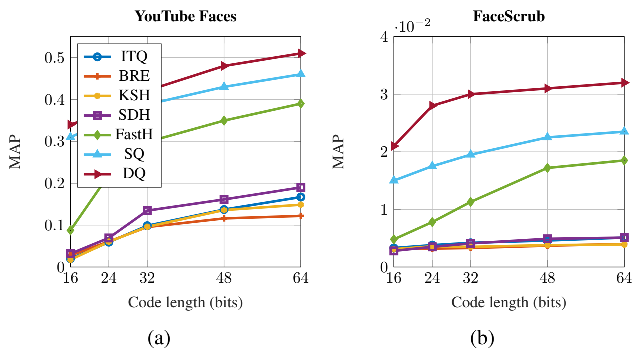

Figure 1: Mean average precision for different lengths of code.

γ for the regularization term and μ for penalizing the the equality constraint term. We set these parameters using a validation set of size 1,000. The best setting of parameters are chosen so that the average search performance in terms of MAP is maximized for the validation set. We preset the p (the dimension of the discrimintative space) to 128 and α to 1, although tuning them may result in better performance. Finally, we choose $h = 256$ to be the dictionary size so that the each subcenter index ($\mathbf{b}_{ij}$) fits into one byte.

3.2. Datasets

To evaluate our method with the proposed evaluation protocols, we require datasets with large number of class labels. Interestingly, face identification benchmarks satisfy this requirement as we often have hundreds of class labels in such datasets. For this reason, we use the following two public face identification benchmarks:

YouTube Faces [15] contains 3,452 videos of 1,595 unique persons. From the videos, 40 faces per person are selected for the experiments which gives us 63800 images.

FaceScrub [9] includes a total of 106,863 face images of 530 celebrities, with almost 200 images per person. It is considered as one of the largest public face datasets.

Following [7], we utilize 236-dimensional LBP feature vectors to represent the visual content.

3.3. Results and Discussions

To evaluate the performance of different techniques, for YouTube Faces dataset, we extract 5 images per person from the videos to form the query set. Therefore, we have 63800 images as the base/training set and 7975 images in the query set for which the average performance is reported. Similarly, for the FaceScrub dataset, we select 5 images per person as the query set and the remaining as the base.

Figure 1 shows the MAP values for different code lengths. It can be seen that DQ achieves the best performance and SQ is the second best among the supervised compact coding techniques for all code lengths. Also, Figure 1 illustrates that quantization based techniques tend to outperform binary hashing techniques. However, this comes at the cost of slightly longer query time (not shown) which is expected as the Hamming distance between binary codes can be computed extremely fast.

Another observation is that the relative difference between the performance of DQ and other techniques is more substantial on the YouTube dataset. Considering that the number of class labels in YouTube dataset is almost triple the FaceScrube, we conjecture that DQ tends to exhibit superior performance specially when the number of class labels is large. However, more empirical evaluations are required to validate this claim.

References

[1] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *TPAMI*, 36(4):744–755, 2014. 2
[2] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2013. 3
[3] Z. Hu, J. Chen, H. Lu, and T. Zhang. Bayesian supervised hashing. In *CVPR*, 2017. 1
[4] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, 2011. 1
[5] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009. 3
[6] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, pages 1963–1970, 2014. 3
[7] J. Lin, Z. Li, and J. Tang. Discriminative deep hashing for scalable face image retrieval. In *IJCAI*, 2017. 4
[8] J. Martinez, S. Zakhmi, H. H. Hoos, and J. J. Little. Lsq++: Lower running time and higher recall in multi-codebook quantization. In *ECCV*, pages 491–506, 2018. 1
[9] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, pages 343–347, 2014. 4
[10] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011. 3
[11] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, pages 3017–3024, 2013. 1
[12] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015. 3
[13] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014. 1
[14] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Supervised quantization for similarity search. In *CVPR*, pages 2018–2026, 2016. 1, 3
[15] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR*, pages 529–534, 2011. 4
[16] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, pages 838–846, 2014. 2