Deep Spherical Quantization for Image Search

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

Abstract

Hashing methods, which encode high-dimensional images with compact discrete codes, have been widely applied to enhance large-scale image retrieval. In this paper, we put forward Deep Spherical Quantization (DSQ), a novel method to make deep convolutional neural networks generate supervised and compact binary codes for efficient image search. Our approach simultaneously learns a mapping that transforms the input images into a low-dimensional discriminative space, and quantizes the transformed data points using multi-codebook quantization. To eliminate the negative effect of norm variance on codebook learning, we force the network to L_2 normalize the extracted features and then quantize the resulting vectors using a new supervised quantization technique specifically designed for points lying on a unit hypersphere. Furthermore, we introduce an easy-to-implement extension of our quantization technique that enforces sparsity on the codebooks. Extensive experiments demonstrate that DSQ and its sparse variant can generate semantically separable compact binary codes outperforming many state-of-the-art image retrieval methods on three benchmarks.

1. Introduction

Nearest neighbor search is one of the fundamental problems in multimedia systems. Given a query point, the goal entails finding the most similar item to the query in a dataset. Accuracy and speed are two key aspects in retrieval systems, however, with explosive growth of high dimensional items such as images, videos and documents on the Internet, most of traditional branch and bound indexing data structures are deemed impractical, mainly because of their query time or memory cost that grow exponentially with the number of dimensions. This has led to a burgeoning field of research, *Approximate Nearest Neighbor* (ANN), that focuses on reducing storage and computational costs with minimal accuracy loss.

ANN problem has witnessed a great amount of research over the past two decades. The state-of-the-art in ANN is mainly focused on *hashing* (compact coding), which aims at encoding high-dimensional media data into short binary codes subject to preserving a given notation of similarity. Binary-valued representation has several advantages, such as being compact to store and faster to compare, making it a suitable fit for large-scale nearest neighbor search. Moreover, for binary strings, one can achieve sublinear query time using hash tables [13, 34] or tree-based indexing data structures [11, 12]. Finding compact binary codes that better respect the given notion of similarity has been the topic of much work over the last two decades during which a rich set of hashing techniques has been proposed. Compact coding techniques are roughly in two streams, categorized by the way they compute distance between encoded items: 1) *Binary hashing* maps high-dimensional input vectors into Hamming space where the distance between two codes can be computed extremely fast using bitwise operators. 2) *Multi-Codebook Quantization* (MCQ) which, analogous to k-means algorithm, partitions the input space into non-overlapping cells and then approximates the distance between two points with the distance between the centers of cell they belong to. The search speed enhancement of MCQ stems from the fact that the distance between cells can be pre-computed and stored in lookup tables.

Not surprisingly, with the dawn of deep learning, most of recent research effort in compact coding has been directed towards using deep networks for producing compact and functional binary codes. Deep hashing methods simultaneously learn the representation and hash coding from raw images. Similarly, deep MCQ has been the topic of study in recent years [4, 17]. Surprisingly enough, although MCQ is a more powerful model as it enables producing many more possible distinct distances, due to the lack of research, its performance in the context of deep supervised compact coding is inferior to state-of-the-art in supervised binary hashing [22].

Most of existing deep supervised MCQ techniques incorporate an unsupervised quantization (usually product quantization (PQ) [18]) on top of the features generated by a deep architecture. Nevertheless, the adopted networks often produce deep features with relatively high norm variance

which adversely affects the quality of quantization [44]. To address this shortcoming, we reformulate the quantization problem by L_2 normalizing deep features to remove norm variance. By exploiting the fact that resulting features lie on a hypersphere, we propose a novel MCQ algorithm that drops the hard orthogonality constraint of product quantization to achieve lower quantization error. Furthemore, to encourage better discriminating performance, inspired by the recently proposed center loss [43], we add a supervised quantization loss term to the final objective function to increase inter-class variance. Finaly, we propose a sparse extension of our quantization algorithm which is necessary for dealing with large codebooks [47]. Comprehensive empirical studies on three standard image retrieval benchmarks testify that DSQ generates compact binary codes which outperform many state-of-the-art methods.

2. Related Work

Existing hashing methods consist of supervised and unsupervised hashing. We refer interested readers to [41] for a comprehensive survey.

Unsupervised hashing methods learn hash functions that map data to binary codes using unlabeled data. Typical learning criteria are reconstruction error minimization [20], preserving local neighborhood [26] and quantization error minimization [15]. Supervised hashing, on the other hand, aims at learning binary codes that are faithful to a given notion of semantic information such as point-wise (class labels) [22, 37, 42], pairwise [2, 3, 6] or triplet labels [24, 33].

Multi-Codebook Quantization. A subclass of unsupervised hashing methods, called Multi-Codebook Quantization (MCQ), is formulated as a quantization problem which aims at approximating vectors with summation of multiple codewords. Formally, let $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ denote the set of n points to be quantized, MCQ is the problem of finding 1) m codebooks (dictionaries) $C_j \in \mathbb{R}^{d \times h}, j \in \{1 \dots, m\}$ each containing h codewords, and 2) the encoding binary vectors $\mathbf{b}_i = [\mathbf{b}_{i1}^T, \dots, \mathbf{b}_{im}^T]^T \in \{0,1\}^{mh \times 1}$, that minimize the quantization error:

$$\sum_{i=1}^{n} \|\mathbf{x}_i - [C_1, \dots, C_m]\mathbf{b}_i\|_2^2 \tag{1}$$

where each subcode $\mathbf{b}_{ij}$ is limited to having only one non-zero entry, $\|\mathbf{b}_{ij}\|_1 = 1$, to ensure only one codeword per codebook is selected. During the query phase, MCQ uses the approximation of each point to estimate the distance between the query $\mathbf{q} \in \mathbb{R}^d$ and each data point:

$$\begin{aligned}\|\mathbf{q} - \mathbf{x}_i\|_2^2 \approx \sum_{i=1}^{m} \|\mathbf{q} - C\mathbf{b}_i\|_2^2 - (m-1)\|\mathbf{q}\|_2^2 \\ + \sum_{t \neq j} (C_t \mathbf{b}_{it})^T C_j \mathbf{b}_{ij}\end{aligned} \tag{2}$$

Given the query, the first term can be efficiently computed using lookup tables that store the distance between the query and each codeword. The second term can be ignored during search time as it is constant for a given fixed query. One of the key features that differentiates MCQ techniques is how they handle the third term. Product quantization (PQ) [18], Cartesian K-means (CKM) [32] and Optimized Product Quantization (OPQ) [14] restrict the codebooks to be mutually orthogonal making the third term equal to zero. Composite Quantization [40], on the other hand, forces it to be a constant which in turn makes the resulting optimization problem hard to solve.

Additive quantization (AQ) [1] and its enhanced extension Local Search Quantization (LSQ) [29] expand $\|\mathbf{q} - \mathbf{x}_i\|_2^2$ based on the inner product of query and codewords but their formulation requires not only approximating the input vector but also its L_2 norm $\|\mathbf{x}_i\|_2^2$. AQ provides two solutions to estimate the norm. The first is to separately quantize the scalar value $\|\mathbf{x}_i\|_2^2$, which results in an additional memory cost that grows linearly with the database size. The other way is to estimate the norm with the codewords which makes the cost of distance computation quadratic in the number of codebooks.

Supervised MCQ. While most of the research in supervised hashing is focused on supervised binary hashing, a handful of studies have been recently proposed on using MCQ in the supervised setting. Supervised MCQ techniques can be, for the most part, described as a combination of supervised loss function and one of the unsupervised MCQ techniques described above. Supervised Quantization (SQ) [42] combines supervised L_2 loss with CQ, however, the resulting optimization problem is hard to solve as it inherits the constant inter-dictionary-element-product constraint from CQ. Deep Quantization Network (DQN) [4] combines a deep architecture and PQ. One shortcoming of DQN is that during codebook optimization, it ignores the supervisory information. SUBIC [17] integrates the one hot-encoding layer in deep neural network which encodes each image with concatenation of one-hot block similar to PQ. However, its sparse property limits its representation capability.

Effect of Norm Variance on MCQ. Recently, Wu *et al*. [44] have shown that norm variance adversely affects the quantization error of unsupervised MCQ techniques, even when the variance is relatively moderate. To address this issue, authors propose to separately scalar quantize the data point norms and then unit normalize the data points before applying PQ. Nevertheless, it is not clear how the quantization budget should be split between the two the quantizers. Also, PQ imposes strong orthogonality on the codebooks which reduces the fidelity of leaned codebooks [1].

We conclude this section by empirically showing the effect of norm variance on the performance of supervised

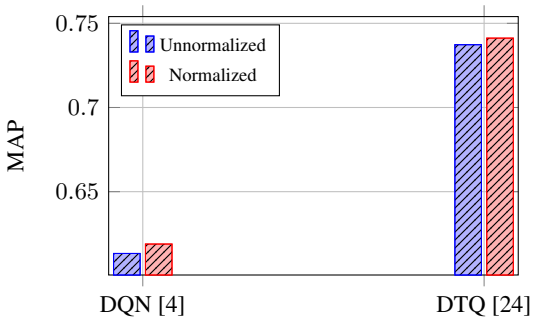

Figure 1: Performance of two supervised MCQ models with and without feature normalization on CIFAR-10 for 64-bit codes.

quantization. To this aim, we run two state-of-the-art supervised MCQ techniques with and without feature normalization on CIFAR-10 dataset (settings of the experiment will be discussed in Section 4). Figure 1 plots the MAP performance of two supervised MCQ techniques and it demonstrates that one can achieve marginal performance gain by simply normalizing the features during training without incurring any additional cost.

3. Proposed Approach

In similarity retrieval, we are given a training set of n points, $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$, with each point associated with a class label, $y_i \in \{1, \dots, l\}$. The goal, given query point $\mathbf{q} \in \mathbb{R}^d$, entails (approximately) finding items in $\mathcal{X}$ that are semantically closest to $\mathbf{q}$ so that the found neighbors share the same class label as $\mathbf{q}$. This paper follows the idea of compact coding techniques that is converting database vectors into compact code and then performing the similarity search in the resulting space which has the advantage of lower memory cost and fast distance computation.

In this paper, we propose to use a deep network that maps the input points into a discriminative space, and simultaneously perform a form of a supervised MCQ on the embedded points to achieve fast retrieval with low computational and storage overhead. To this aim, we define a loss function comprising four terms, softmax loss, center loss, quantization loss, and discriminative loss each of which will be discussed in the following.

3.1. Softmax and Center loss

In deep retrieval systems, obtaining a robust and discriminative representation is crucial for achieving good performance. Usually, this is achieved by applying the softmax loss to the representation layer of the network. However, the resulting features optimized with the supervision of softmax loss are often not discriminative enough as the softmax loss only focuses on finding a decision boundary that separates different classes without considering the intra-class compactness which is crucial to the accuracy of nearest neighbor search [16, 43].

To increase the intra-class variations while keeping the features of different classes separable, we adopt the state-of-the-art *center loss* [43] on top of the softmax loss.

Let $f(\cdot;\theta) : \mathbb{R}^d \rightarrow \mathbb{R}^p$, with $p \ll d$, denote the feed-forward network that embed the input vectors into p-dimensional deep features, also let $\mathbf{z}_i$ denote deep feature representation of input $\mathbf{x}_i$, $\mathbf{z}_i = f(\mathbf{x}_i;\theta)$, then, the center loss is defined as:

$$L_C = \sum_{i=1}^{n} \|\mathbf{z}_i - \phi_{y_i}\|_2^2 \tag{3}$$

where y_i is the classes label associated with $\mathbf{z}_i$ and ϕ_{y_i} denotes the y_i-th class center of deep features. Intuitively, center loss learns a center for the features of each class and meanwhile aims at pulling the deep features of the same class close to its corresponding center. It has been shown that joint supervision of softmax loss and center loss can produce significantly better discriminative deep features [43].

3.2. Quantization loss

We constrain the deep features to live on a p-dimensional unit hypersphere, *i.e.* $\|f(\mathbf{x};\theta)\|_2 = 1$. Other than decreasing the intra-class variability of deep features [39], there are two advantages in normalizing feature vectors: 1) norm variance is strictly zero, and 2) Euclidean nearest neighbor search is equivalent to Maximum Inner Product Search (MIPS) as for unit norm vectors we have $\|\mathbf{q} - \mathbf{x}\|_2^2 = 2 - 2\mathbf{q}^T\mathbf{x}$.

The main benefit of dealing with MIPS is that, unlike Euclidean distance (see (2)), inner product naturally satisfies the distributive law, that is $\langle \mathbf{q}, \sum_j \mathbf{t}_j \rangle = \sum_j \langle \mathbf{q}, \mathbf{t}_j \rangle$. MCQ works well in large part due to the fact that it permits the distance between query and a quantized point to be computed as the summation of partial distances between query and selected codewords. Given the query, the distances between query and all codewords are stored in query-specific lookup tables and then used to calculate the distance between query and all quantized points. However, to make Euclidean distance satisfy the distributive law, we either need to enforce strong [18, 32]/weak [46, 47] orthogonality constraints over the codewords of different dictionaries which reduces the fidelity of model and often leads to non-convex optimization, or we have to store the inner product between the all codewords in lookup table [1, 29] which increases storage cost and distance computation time.

To reduce the approximation error of MIPS, we need to minimize the *distance reconstruction error* of MCQ. Since the Euclidean distance on the unit sphere is equal to the

negative dot product plus a constant, distance reconstruction error can be rewritten as:

$$\mathbb{E}_{\mathbf{q}\sim P(\mathbf{q})}\Big[\sum_{i=1}^{n}|\langle \mathbf{z}_q, \mathbf{z}_i\rangle - \langle \mathbf{z}_q, \bar{\mathbf{z}}_i\rangle|\Big] = \mathbb{E}_{\mathbf{q}\sim P(\mathbf{q})}\Big[\sum_{i=1}^{n}\langle \mathbf{z}_q, \mathbf{z}_i - \bar{\mathbf{z}}_i\rangle\Big] \leq \sum_{i=1}^{n}\|\mathbf{z}_i - \bar{\mathbf{z}}_i\|_2 \tag{4}$$

where $\bar{\mathbf{z}}_i$ denotes the approximation of $\mathbf{z}_i$ using MCQ and $\mathbf{z}_q = f(\mathbf{q};\theta)$.

This suggests that the search accuracy directly depends on the quantization error; low quantization error leads to high search accuracy.

Therefore, the cost function we aim to optimize is the quantization loss:

$$\begin{aligned} L_Q(\{C_j\},\{\mathbf{b}_i\}) = \quad & \sum_{i=1}^{n}\|\mathbf{z}_i - [C_1,\dots,C_m]\mathbf{b}_i\|_2^2 \\ & \mathbf{b}_i = [\mathbf{b}_{i1}^T,\dots,\mathbf{b}_{im}^T]^T \\ & \mathbf{b}_{ij}\in\{0,1\}^h, \|\mathbf{b}_{ij}\|_1 = 1 \\ & j = 1,\dots m \end{aligned} \tag{5}$$

The benefit of such a simple formulation, in comparison to those that enforce multiple constraints on the codewords [18, 32, 46] are multi-fold; it causes a straightforward optimization procedure and also less implementation overhead.

3.3. Discriminative Dictionary Learning

Finally, we also incorporate the supervisory information during quantization procedure. In particular, we encourage the quantized points to be closer to their centers. To achieve this goal, we use the following loss:

$$L_D = \sum_{i=1}^{n}\|\phi_{y_i} - C\mathbf{b}_i\|_2^2 \tag{6}$$

Intuitively, (6) penalizes the cases where the point $\bar{\mathbf{z}}_i$ is not assigned to the clusters that are close to ϕ_{y_i}.

The overall loss for training model takes the form:

$$L = L_{softmax} + \alpha L_Q + \lambda L_C + \gamma L_D \tag{7}$$

where α, λ and γ are the hyper-parameters that control the effect of each term.

3.4. Optimization

The objective function composes of four sets of learnable parameters, the parameters of the deep network θ, the centers ϕ_{y_i}s, the codewords in matrix C, the codeword assignment matrix B. We use alternative optimization to solve the problem with each iteration updating one set of parameters while fixing others.

Updating θ. With C, ϕ_{y_i}s, and B fixed, the parameters of the network are updated through back-propagation as all of the terms in the loss are differentiable.

Updating Φ. We follow a similar procedure to [43] for updating the centers. In particular, to avoid large perturbation caused by few mislabelled instances, we use a learning rate parameter ζ for training the centers:

$$\phi_{y_i}^{t+1} = \phi_{y_i}^{t} - \zeta\Delta\phi_{y_j} \tag{8}$$

$$\Delta\phi_{y_j} = \frac{\sum_{i=1}^{n}\mathbb{1}(y_i = j)\cdot[\lambda(\phi_{y_i} - \mathbf{z}_i) + \gamma(\phi_{y_i} - C\mathbf{b}_i)]}{1 + \sum_{i=1}^{n}\mathbb{1}(y_i = j)} \tag{9}$$

where $\mathbb{1}(condition)$ equals 1 if the condition is satisfied and 0 otherwise. Ideally, the centers should be updated in each iteration based on the whole training set which would be extremely costly. To reduce the cost, the update is performed on the mini-batches.

Updating C. Given B, ϕ_{y_i}s and θ fixed, the resulting optimization problem is:

$$\alpha\|Z - CB\|_2^2 + \gamma\|\Phi - CB\|_2^2 \tag{10}$$

where $Z = [\mathbf{z}_1,\dots,\mathbf{z}_n]$, $B = [\mathbf{b}_1,\dots,\mathbf{b}_n]$, and $\Phi = [\phi_{y_1},\dots,\phi_{y_n}]$. This is a quadratic function in C and therefore a closed-form solution exists:

$$C = \frac{1}{\alpha+\gamma}(\alpha Z + \gamma\Phi)B^T(BB^T)^{-1} \tag{11}$$

It is easy to observe that the optimization problem decomposes over each of the p dimensions. Thus, we can reduce the computational cost by solving p least square problem each with mh variables.

$$\min_{C^{(t)}} \alpha\|Z^{(t)} - C^{(t)}B^{(t)}\|_2^2 + \gamma\|\Phi^{(t)} - C^{(t)}B^{(t)}\|_2^2 \quad \forall \quad t = 1,\dots,p \tag{12}$$

Each of the p problems is a least squares problem with a closed form solution. Online learning algorithms can also be leveraged for acceleration [28].

Updating B. Given θ, ϕ_{y_i} and C fixed, optimizing binary matrix B, known as encoding phase, has been historically identified as the bottleneck of MCQ [1, 29].

It can be seen that the composition indicator vector $\mathbf{b}_i$ is independent of all other vectors $\{\mathbf{b}_t\}_{t\neq i}$. Thus, the optimization problem with respect to B can be decomposed into n independent subproblems:

$$\begin{aligned} \min_{\mathbf{b}_i} \quad & \alpha\|\mathbf{z}_i - C\mathbf{b}_i\|_2^2 + \gamma\|\phi_{y_i} - C\mathbf{b}_i\|_2^2 \\ & \mathbf{b}_i = [\mathbf{b}_{i1}^T,\dots,\mathbf{b}_{im}^T]^T \\ & \mathbf{b}_{ij}\in\{0,1\}^h, \|\mathbf{b}_{ij}\|_1 = 1 \\ & i = 1,\dots,n \quad j = 1,\dots,m \end{aligned} \tag{13}$$

The problem is essentially a high-order Markov Random Field (MRF) problem which is NP-hard. Following [29], we use Stochastic Local Search (SLS) method to optimize $\mathbf{b}_i$. The idea of SLS for escaping local minima is to iteratively alternate between a local search procedure, and a randomized pertubation to the current solution. For the local search, we again use alternative optimization technique. Given $\{\mathbf{b}_{ij}\}_{j\neq t}$ fixed, $\mathbf{b}_{it}$ is updated by exhaustively checking all codewords of C_j and finding the element that minimizes the objective function in (13). For the perturbation procedure of SLS, we randomly choose k codes by sampling from the uniform distribution $\mathcal{U}(1, m)$. The selected codes are perturbed by setting each of them to a uniformly selected random value between 1 and h. The resulting perturbed solution is then accepted as the starting point of the next local search procedure. Although this procedure is computationally demanding, it can be accelerated using GPU implementation [30, 31], making encoding even faster than codebook learning.

3.5. Asymmetric Distance Computation

Given the query, the search process starts by embedding the query using the trained network, $\mathbf{z}_q = f(\mathbf{q}; \theta)$. Then, the inner product between $\mathbf{z}_q$ and all codewords are stored in $m \times h$ query-specific lookup table. Finally, inner product between query and all database vector is approximated with:

$$\langle \mathbf{z}_q, \mathbf{z}_i \rangle \approx \sum_{j=1}^{m} \langle \mathbf{z}_q, C_j \mathbf{b}_{ij} \rangle \tag{14}$$

Therefore, computing the inner product between query and each database item takes $O(m)$ lookups and $O(m)$ addition operations (same as PQ), plus the time required to embed the query into the deep feature space.

3.6. Sparse Codebook Learning

In sparse codebook learning, the optimization problem is augmented with sparsity constraint on the codewords. The key advantage of sparse coodebooks is that the distance between the query and every codeword can be computed efficiently using sparse vector manipulations. This is practically important as for large codebooks, with many codewords, the time required for online construction of lookup tables become non-negligible. Zhang *et al*. [47] have shown that sparse codewords can increase the search speed up to 30%. As the name suggests, the Sparse Composite Quantization (SCQ) technique proposed in [47] adds sparsity constraint to the CQ [46] formulation and uses coordinate descent to solve the optimization problem. However, CQ itself involves a hard optimization problem and adding the sparsity constraint makes the problem even harder.

In contrast, in our formulation, codebook optimization reduces to a linear regression problem, thus adding the sparsity constraint changes the objective to a regularized quadratic problem. In particular, using straightforward algebraic manipulations (10) can be rewritten as:

$$\begin{aligned}(\alpha+\gamma)\|\frac{\alpha Z+\gamma\Phi}{\alpha+\gamma}-CB\|_2^2 - \frac{\|\alpha Z+\gamma\Phi\|_2^2}{\alpha+\gamma} \\ + \alpha\|Z\|_2^2+\gamma\|\Phi\|_2^2\end{aligned} \tag{15}$$

Since only the first term depends on C, we can write the objective function of sparse quantization as:

$$\min_C \|\frac{\alpha Z+\gamma\Phi}{\alpha+\gamma}-CB\|_2^2 \quad s.t. \quad \|C\|_0 \leq \epsilon \tag{16}$$

The resulting optimization is non-convex because of L_0 regularization term. Commonly, such problems are relaxed by replacing L_0 norm with convex L_1 norm. Therefore, our final objective function for learning sparse codebooks is defined as:

$$\min_C \|\frac{\alpha Z^T+\gamma\Phi^T}{\alpha+\gamma}-B^T C^T\|_2^2 \quad s.t. \quad \|C\|_1 \leq \epsilon \tag{17}$$

which is essentially a linear regression problem with L_1 norm regularization on the coefficients, known as Lasso in the statistical literature. It can be efficiently solved using a wide range of heavily-optimized off-the-shelf Lasso solvers such as feature-sign search [21] or SPGL1 solver [38].

4. Experiments

In this section, we gauge the performance of the proposed supervised quantization approach by comparing it with the state-of-the-art against three different datasets..

4.1. Datasets and Evaluation

We conduct experiments on three standard datasets: CIFAR-10 [19], NUS-WIDE [8] and ImageNet [9].

CIFAR-10 dataset consists of 60,000 32×32 color images evenly divided into 10 categories. We follow the official split of the datasets and use 50K images as the training set and 10k images as the query set.

NUS-WIDE is a set of 269,648 images collected from Flickr. This is a multi-label dataset where each image is associated with one or multiple labels from a given 81 concepts. Following [37, 42], we collect 193,752 images that are from the 21 most frequent labels for evaluation, including *sky, clouds, person, water, animal, grass, building, window, plants, lake, ocean, road, flowers, sunset, relocation, rocks, vehicles, snow, tree, beach, and mountain*. For each label, we randomly sample 100 images as query points and the remaining images form the training set.

The dataset ILSVRC 2012, named as ImageNet in this paper, contains over 1.2 million images covering 1,000 categories. Following the settings in [5, 7], we select 100 categories and use images associated with them in the provided

training set and the validation set as the training and the query sets, respectively.

Parameter setting. There are trade-off parameters in the objective function (7): α for quantization loss, λ for center loss and γ for discriminative loss. We select parameters via validation. In particular, we choose a subset of the training set (same size as the query set), and the best parameters are chosen so that the average performance in terms of MAP is maximized against the validation set. We fix ζ to 0.5 and k to 4.

Following almost all MCQ techniques [1, 32, 47], we choose $h = 256$ to be the codebook size, so that each subindex fits into one byte of memory. This let us store B as a $m \times n$ `uint8` matrix. We vary $m = \{2, 4, 6, 8\}$ such that $m \log_2 h$ is equal to the desired bit-rates which are $\{16, 32, 48, 64\}$.

Experimental settings. We use the raw images as the input for all deep methods, but the images are resized to fit the input of the adopted model. For fairness of comparison, for all deep compact coding methods here, we use Alexnet as the core architecture. To reduce the size of deep features, we add a fully connected layer to the network which transforms the output of the network into a 256-dimensional feature space, thus $p = 256$. We do not tune the size of feature space for saving time while we think that tuning it might yield better performance. The L_2 normalization is performed on the 256-dimensional deep features using a L_2 normalize layer [35].

We fine-tune layers conv1–fc7 copied from the AlexNet model pre-trained on ImageNet and train the last layer which maps the feature layer via back-propagation. As the last layer is trained from scratch, we set its learning rate to be 10 times that of the other layers. We use mini-batch stochastic gradient descent (SGD) with 0.9 momentum as the solver, and cross-validate the learning rate from 10^{-5} to 10^{-2} with a multiplicative step-size $\sqrt{10}$. We also fix the mini-batch size of images as 128 and the weight decay parameter as 0.0005. Following [29], we use SPGL1 as the lasso solver for the sparse extension of our algorithm [38]. For non-deep methods, we extract the outputs of the layer 'fc7' in the deep model [10] as input features.

Methods. We compare DSQ with a wide range of supervised compact coding methods including binary hashing methods: KSH [27], ITQ [15], SDH [37], CNNH [45], DPSH [23], DSH [25], HashNet [5], and supervised quantization techniques: SQ [42], SUBIC [17], DQN [4] and DTQ [24]. We implemented SQ in Python as its source code is not available at the time of writing this paper. We tried our best to be faithful to the experimental settings of the paper [42]. Other techniques are executed using the implementation generously provided by the authors.

4.2. Results

Single domain retrieval. Single-domain retrieval is the main experimental benchmark in the supervised binary hashing literature in which the query and training items belong to the same set of class labels. To evaluate performance of different techniques, we adopt the widely used Mean Average Precision (MAP). We report the results of MAP@5000 and MAP@1000 for NUS-WIDE and ImageNet datasets respectively. Table 1 shows the single-domain retrieval performance of DSQ against a wide-range of techniques. The observation is that our proposed method consistently delivers the best performance for different length of codes. We attribute the performance improvement to the proposed loss that aims at jointly preserving similarity information and controlling the quantization error. Also, dropping the orthogonality constraint increases the fidelity of codebooks which in turn reduces the approximation error of nearest neighbor search. Finally, back-proping the proposed supervised quantization loss can remarkably enhance the quantizibilty of the deep representation.

Figure 2 also shows the performance of different techniques in terms of the precision-recall curves for 64-bit codes. From the curves, we can observe that DSQ delivers higher precision than the state-of-the-art compact coding methods at the same recall rate. This shows that DSQ is also favourable for precision-oriented retrieval systems. Although the query time comparison is not presented here due to space limit, we observed that all deep MCQ techniques in this study exhibit similar query time mainly because they adopt the same core architecture (AlexNet). However, binary hashing techniques are often faster than deep MCQ as they incorporate Hamming distance to compare binary codes.

Sparse coding. We also show the performance of sparse extension of DSQ. To the best of our knowledge, sparse DSQ is the first attempt to explore supervised sparse multi-codebook quantization for semantic similarity search. Nevertheless, we compare our technique with two unsupervised sparse quantization techniques, SCQ [47] and SLSQ [29] applied to the deep features of the 'fc7' layer of the deep model in [10].

Following [46], we evaluate the sparse version of our algorithm using two degrees of sparsity: SDSQ1 with $\|C\|_0 \leq \epsilon = h \cdot p$ and SDSQ2 with $\|C\|_0 \leq \epsilon = h \cdot p + p^2$. Since the former criterion imposes a harder sparsity constraint on the codebooks, we would naturally expect to achieve lower search accuracy but better query time. We compare against SCQ1 and SCQ2 from [47] and SLSQ1 and SLSQ2 from [29].

Figure 3 shows the performance of different techniques against three different datasets. Again, in this scenario, we observe that sparse DSQ comfortably outperforms the baselines with a large margin mainly because sparse DSQ

	CIFAR-10				NUS-WIDE				ImageNet			
Method	16	32	48	64	16	32	48	64	16	32	48	64
KSQ	0.3216	0.3285	0.3371	0.3384	0.4061	0.4182	0.4264	0.4436	0.1620	0.2818	0.3422	0.3934
ITQ	0.2412	0.2432	0.2482	0.2531	0.5573	0.5932	0.6128	0.6166	0.3115	0.4632	0.5223	0.5446
SDH	0.4199	0.4301	0.4392	0.4465	0.5342	0.6282	0.6298	0.6335	0.2729	0.4521	0.5329	0.5893
CNNH	0.5373	0.5421	0.5765	0.5780	0.6221	0.6233	0.6321	0.6372	0.2888	0.4472	0.5328	0.5436
DPSH	0.6367	0.6412	0.6573	0.6676	0.7015	0.7126	0.7418	0.7423	0.3226	0.5436	0.6217	0.6534
DSH	0.6192	0.6565	0.6624	0.6713	0.7181	0.7221	0.7521	0.7531	0.3428	0.5500	0.6329	0.6645
HashNet	0.6857	0.6923	0.7183	0.7187	0.7331	0.7551	0.7622	0.7762	0.5016	0.6219	0.6613	0.6824
DTQ	0.7037	0.7191	0.7319	0.7373	0.7511	0.7812	0.7886	0.7892	0.5128	0.6123	0.6727	0.6916
SUBIC	0.6555	0.6789	0.6854	0.7014	0.7021	0.7131	0.7555	0.7568	0.5547	0.5597	0.6462	0.6622
SQ	0.6212	0.6438	0.6545	0.6578	0.7126	0.7138	0.7303	0.7423	0.3865	0.5586	0.6279	0.6618
DQN	0.5979	0.6097	0.6099	0.6133	0.6913	0.7121	0.7471	0.7562	0.5065	0.6205	0.6669	0.6912
DSQ	**0.7212**	**0.7346**	**0.7418**	**0.7589**	**0.7785**	**0.7899**	**0.7918**	**0.7988**	**0.5769**	**0.6541**	**0.6800**	**0.6940**

Table 1: Single-domain category retrieval performance of DSQ versus the state-of-the-art with 16, 32, 48 and 64 bit codes.

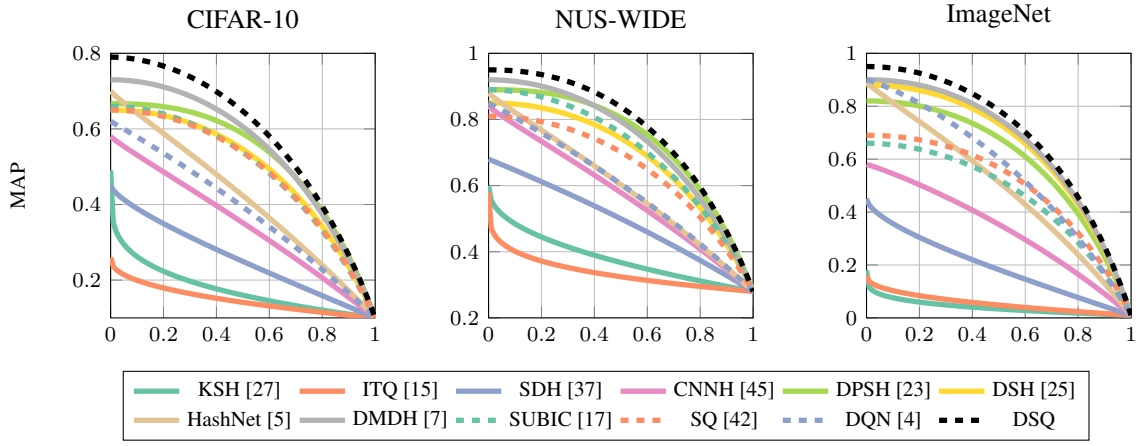

Figure 2: Precision-recall curves on the CIFAR-10, NUS-WIDE and ImageNet datasets for 64-bit codes.

jointly optimizes the quantization error while preserving the semantic similarity and satisfying the sparsity constraint, whereas the other benchmarks separately apply unsupervised sparse quantization, which merely minimizes the quantization error.

Cross-domain retrieval. To further evaluate our supervised quantization method, we follow an alternative evaluation protocol from [36] wherein the model learned on a given set of training classes is tested on a new, disjoint set of test classes. This protocol is used to show how each method is capable of preserving the semantic information of certain classes implicitly even if the class samples are not included in the training set.

Toward this aim, we partition the samples based on their class labels such that 70% of the labels belong to the training and the remaining labels are used to form the base and query set. Note that in this scenario the training set is used to optimize the parameters of the model. Once learning is completed, the training set is removed and the items of base set are mapped into compact codes using the trained model. Finally, the average performance over the query set is reported. We use 80% of the samples with unseen classes as the training set and the rest as the query set. This process is repeated 5 times with random class splits and the average results is reported. For this setting, during the encoding phase, we drop the L_C term from loss because the trained centers do not correspond to any of the labels in the base set. Similarly, the regression loss term in SQ [42] is dropped during encoding as it directly depends on the class labels of training set.

Table 2 demonstrates the results of this experiment which shows the superiority of DSQ for different lengths of code. We also observe that the MAP performance of methods are generally higher than that of the previous protocol since there is less variation in the base set consisting of only 3 classes and fewer samples to retrieve from. Also the rank of techniques is different from the single domain experiments. For example, SUBIC exhibits the closest performance to DSQ whereas in single-domain setting DTQ is the closest.

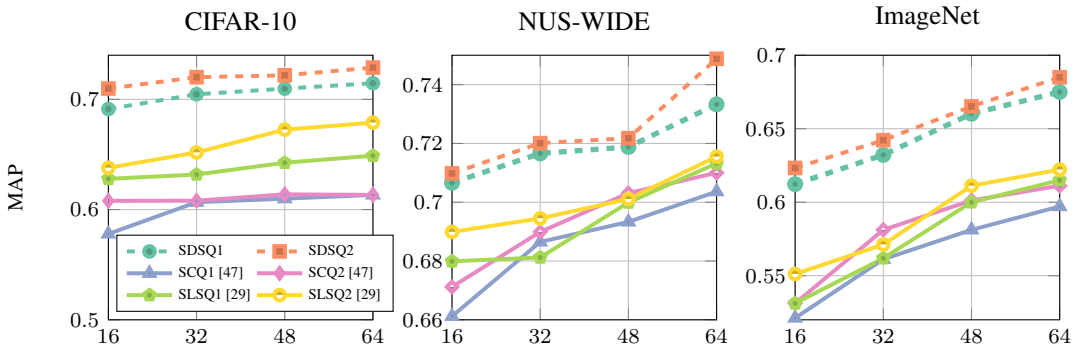

Figure 3: Mean Average Precision performance of different sparse quantization techniques against three datasets.

Method	16	32	48	64
CNNH	0.6241	0.6456	0.6478	0.6491
DPSH	0.6894	0.7134	0.7198	0.7256
HashNet	0.7826	0.7941	0.7989	0.8010
SUBIC	0.7832	0.7931	0.8032	0.8077
DSH	0.7316	0.7388	0.7437	0.7456
SQ	0.7112	0.7126	0.7319	0.7389
DQN	0.7562	0.7612	0.7649	0.7655
DTQ	0.7525	0.7685	0.7700	0.7895
DSQ	**0.7944**	**0.8165**	**0.8195**	**0.8218**

Table 2: Mean Average Precision performance of different techniques for the task of cross domain performance on CIFAR-10.

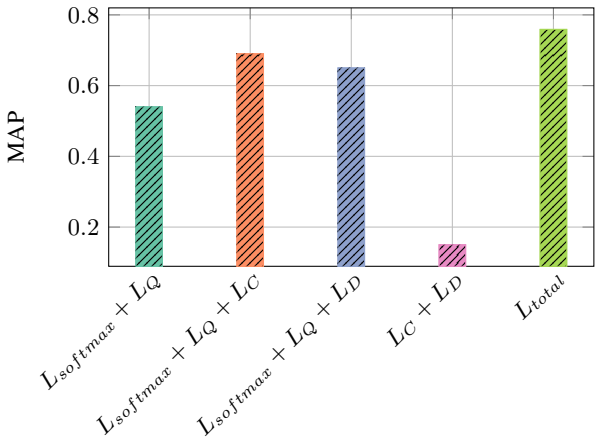

Figure 4: Difference in MAP, when different loss components are excluded from DSQ objective function. The experiments are conducted on 64-bit codes of CIFAR-10 dataset.

4.3. Ablation Study

We also perform an ablation study to showcase the contribution and importance of loss function components on the final performance of the model by empirically comparing different variants of DSQ. We evaluate this experiment across different models to understand the sensitivity of DSQ to different terms: 1) $L_{softmax} + L_Q$, 2) $L_{softmax} + L_Q + L_C$, 3) $L_{softmax} + L_Q + L_D$, and 4) $L_C + L_D$. For each model, the coefficients of different terms are again tuned using cross validation and the average performance of model for 64-bit codes against CIFAR-10 dataset is reported in Figure 4.

The first observation is that all of the loss components contribute in improving MAP. Also, the plot indicates the importance of softmax loss. This is due to the fact that the softmax loss is the only term in the objective function that uses that class labels to force the deep features of different classes staying apart, without it, the resulting loss function degrades all inputs points to be projected onto a single point. The figure also demonstrates considerable contribution of discriminative loss, L_D, showing the effectiveness of our framework in incorporating semantic information during quantization.

5. Conclusion

In this paper, we propose a deep supervised quantization technique for efficient and fast image retrieval. By incorporating L_2 normalized features, we propose a simple yet efficient supervised MCQ algorithm for encoding unit normalized data points with similarity preserving binary codes. We also show that our algorithm can be easily extended to accommodate sparsity constraint in the codebooks which is necessary for learning large-scale codebooks. Comprehensive experiments justify that DSQ and its sparse extension generate compact binary codes that yield state-of-the-art retrieval performance on three standard benchmarks, namely CIFAR-10, NUS-WIDE, and ImageNet.

References

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

[2] Y. Cao, B. Liu, M. Long, J. Wang, and M. KLiss. Hashgan: Deep learning to hash with pair conditional wasserstein gan. In *CVPR*, pages 1287–1296, 2018.

[3] Y. Cao, M. Long, B. Liu, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *CVPR)*, 2018.

[4] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[5] Z. Cao, M. Long, J. Wang, and S. Y. Philip. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5609–5618, 2017.

[6] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou. Deep hashing via discrepancy minimization. In *CVPR*, 2018.

[7] Z. Chena, X. Yuana, J. Lua, Q. Tiand, and J. Zhoua. Deep hashing via discrepancy minimization. In *CVPR*, pages 6838–6847, 2018.

[8] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*, page 48, 2009.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[11] S. Eghbali, H. Ashtiani, and L. Tahvildari. Online nearest neighbor search in binary space. In *ICDM*, pages 853–858, 2017.

[12] S. Eghbali, H. Ashtiani, and L. Tahvildari. Online nearest neighbor search using hamming weight trees. *IEEE Trans. PAMI*, 2019.

[13] S. Eghbali and L. Tahvildari. Fast cosine similarity search in binary space with angular multi-index hashing. *IEEE Trans. on Knowledge and Data Engineering*, 31(2):329–342, 2019.

[14] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Trans. PAMI*, 36(4):744–755, 2014.

[15] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. PAMI*, 35(12):2916–2929, 2013.

[16] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai. Triplet-center loss for multi-view 3d object retrieval. *arXiv preprint arXiv:1803.06189*, 2018.

[17] H. Jain, J. Zepeda, P. Perez, and R. Gribonval. Subic: A supervised, structured binary code for image search. In *ICCV*, Oct 2017.

[18] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, 2011.

[19] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[20] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.

[21] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2007.

[22] Q. Li, Z. Sun, R. He, and T. Tan. Deep supervised discrete hashing. In *NIPS*, pages 2482–2491, 2017.

[23] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016.

[24] B. Liu, Y. Cao, M. Long, J. Wang, and J. Wang. Deep triplet quantization. *MM*, 2018.

[25] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016.

[26] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[27] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[28] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, pages 689–696, 2009.

[29] J. Martinez, J. Clement, H. H. Hoos, and J. J. Little. Revisiting additive quantization. In *ECCV*, pages 137–153, 2016.

[30] J. Martinez, H. H. Hoos, and J. J. Little. Solving multi-codebook quantization in the gpu. In *ECCV*, pages 638–650, 2016.

[31] 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.

[32] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, pages 3017–3024, 2013.

[33] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, pages 1061–1069, 2012.

[34] M. Norouzi, A. Punjani, and D. J. Fleet. Fast exact search in hamming space with multi-index hashing. *IEEE Trans. PAMI*, 36(6):1107–1119, 2014.

[35] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[36] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou. How should we evaluate supervised hashing? In *ICASSP*, pages 1732–1736, 2017.

[37] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[38] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[39] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018.

[40] J. Wang and T. Zhang. Composite quantization. *IEEE Trans. PAMI*, 2018.

[41] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *IEEE Trans. PAMI*, 40(4):769–790, 2018.

[42] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Supervised quantization for similarity search. In *CVPR*, pages 2018–2026, 2016.

[43] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016.

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

[45] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, page 2, 2014.

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

[47] T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Sparse composite quantization. In *CVPR*, pages 4548–4556, 2015.