Learning Deep Unsupervised Binary Codes for Image Retrieval

Junjie Chen[1], **William K. Cheung**[1] and **Anran Wang**[2]
[1] Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
[2] Institute for Infocomm Research, A*STAR, Singapore
{csjjchen, william}@comp.hkbu.edu.hk, wang_anran@i2r.a-star.edu.sg

Abstract

Hashing is an efficient approximate nearest neighbor search method. It has been widely adopted for large-scale multimedia retrieval. While supervised learning is popular for the data-dependent hashing, deep unsupervised hashing methods can learn non-linear transformations for converting multimedia inputs to binary codes without label information. Most of the existing deep unsupervised hashing methods make use of a quadratic constraint for minimizing the difference between the compact representations and the target binary codes, which inevitably causes severe information loss. In this paper, we propose a novel deep unsupervised method called DeepQuan for hashing. The DeepQuan model utilizes a deep autoencoder network, where the encoder is used to learn compact representations and the decoder is for manifold preservation. To contrast with the existing unsupervised methods, DeepQuan learns the binary codes by minimizing the quantization error through product quantization. Furthermore, a weighted triplet loss is proposed to avoid trivial solutions and poor generalization. Extensive experimental results on standard datasets show that the proposed DeepQuan model outperforms the state-of-the-art unsupervised hashing methods for image retrieval tasks.

1 Introduction

With the explosive growth of large volume and high dimensional multimedia data like images and videos, efficient large-scale visual retrieval has received growing attention [Li *et al.*, 2015; Gong *et al.*, 2013]. Hashing, one of the approximate nearest neighbors (ANN) retrieval methods [Andoni and Indyk, 2006], has been widely used in many applications such as image retrieval, due to its computation efficiency and low storage cost. It maps high dimensional data into compact binary codes, where semantically similar items can be indexed by similar binary codes. Existing hashing methods can roughly be divided into two categories: data-independent hashing and data-dependent hashing [Cao *et al.*, 2016; Gong *et al.*, 2013]. It has been shown that the data-dependent methods achieve better performance than the data-independent ones, e.g. Locality Sensitive Hashing(LSH) [Datar *et al.*, 2004]. In this paper, we focus on data-dependent methods to learn binary codes for efficient image retrieval.

Many data-dependent hashing methods have been proposed to learn efficient binary codes for the retrieval task [Cao *et al.*, 2016; Gong *et al.*, 2013; Li *et al.*, 2015; Liu *et al.*, 2011; Zhang *et al.*, 2017; Duan *et al.*, 2017; Erin Liong *et al.*, 2015; Lin *et al.*, 2016; Huang *et al.*, 2017; Do *et al.*, 2016]. These hashing methods can be further divided into supervised hashing and unsupervised hashing. While supervised methods make use of provided labels to mitigate the gap between high-level semantic and low-level feature representation of items, the performance is always hindered by the availability of supervision information. This is especially challenging when deep models are considered as a large amount of labeled data is typically needed for the training. As it is much easier to collect a high volume of unlabeled data, deep unsupervised hashing learning methods become attractive. Deep unsupervised hashing learns multiple hierarchical transformations to capture the nonlinear manifold structure of data [Erin Liong *et al.*, 2015]. While some kernel-based methods [Liu *et al.*, 2012] have been developed for the hashing task, they are however not scalable.

Among the existing deep unsupervised hashing methods proposed for image retrieval [Duan *et al.*, 2017; Erin Liong *et al.*, 2015; Lin *et al.*, 2016; Huang *et al.*, 2017], most of them make use of a simple quadratic constraint to minimize the difference between the compact representations and the target binary codes during the training, which inevitably causes severe information loss and thus produces suboptimal binary codes. Also, as discussed in [Cao *et al.*, 2016; Ge *et al.*, 2014], to quantize all input vectors effectively with vector quantization (VQ) [Gray, 1984], the data should exhibit a clustering structure. Hence it is also important to learn the image representations so that such a clustering structure can be resulted.

In this paper, we propose an unsupervised deep hashing model called DeepQuan (*Deep Quan*tization) to address the aforementioned problems. Our contributions can be summarized as follows:

- Instead of forcing the outputs of the deep network to be binary using a quadratic constraint, we propose to generate binary codes with product quantization (PQ) [Jegou *et al.*, 2011; Ge *et al.*, 2014]. To the best of our knowl-

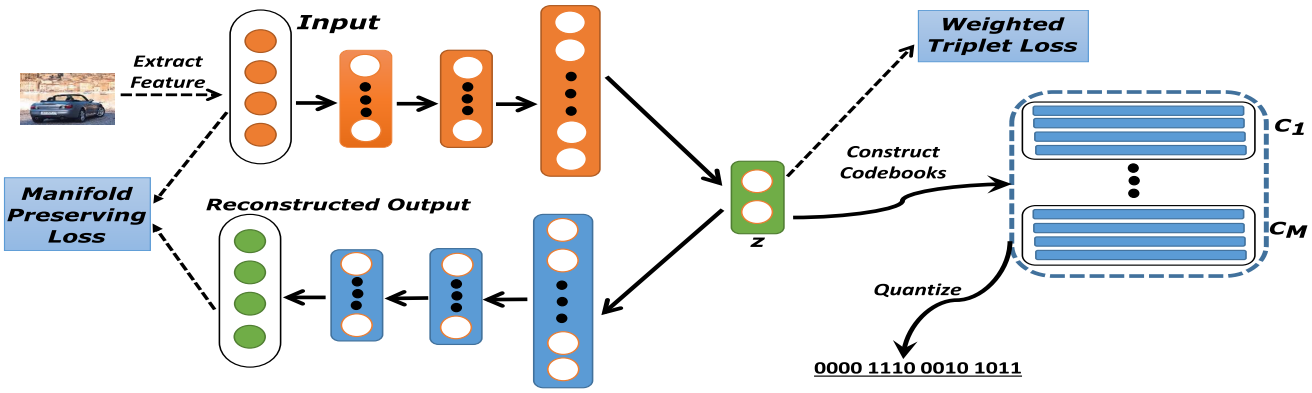


Figure 1: The framework of the proposed DeepQuan model, which is built upon a deep autoencoder. All the layers are fully connected. A weighted triplet loss and a manifold preserving loss are employed to learn discriminative compact representations for better quantization.

edge, the proposed model DeepQuan is the first attempt to incorporate PQ into a deep model to learn binary code in an unsupervised way. This model is built upon a deep autoencoder, and an alternating optimization method is proposed for the model learning. By optimizing the objective function, more discriminative feature representations (*i.e.* a more discriminative clustering structure) can be obtained so that PQ can quantize the feature vectors more effectively.

- To avoid trivial solutions and poor generalization, we propose a novel weighted triplet loss to learn more discriminative features and demonstrate its effectiveness empirically.
- Experimental results on two standard datasets (*i.e.* CIFAR-10 and MNIST) show that DeepQuan outperforms the existing deep unsupervised methods and achieves the state-of-the-art performance in the image retrieval application.

The rest of the paper is organized as follows. In Sec. 2, we introduce some related work. Details of the proposed DeepQuan are described in Sec. 3 The experimental results and discussion are presented in Sec. 4. Sec. 5 concludes the paper with future research directions.

2 Related Work

Unsupervised hashing methods exploits the intrinsic manifold structure of the input data to achieve the hashing objective. For example, Spectral Hashing (SH) [Weiss *et al.*, 2009] was proposed to learn the hash codes by solving a spectral graph partitioning problem. Iterative Quantization (ITQ) obtains the hash codes by imposing an orthogonal transformation matrix to minimize the quantization loss [Gong *et al.*, 2013]. K-means Hashing (KMH) tries to index every quantized codeword with a binary code so that the distance between them is minimized [He *et al.*, 2013].

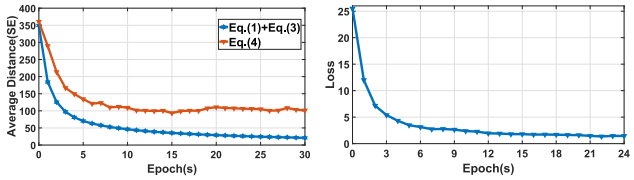
With the recent progress of deep learning [Simonyan and Zisserman, 2014], many deep hashing methods have been proposed. Some deep supervised hashing methods like Deep Pairwise Supervised Hashing (DPSH) [Li *et al.*, 2015] and Deep Quantization Network (DQN) [Cao *et al.*, 2016] have achieved the state-of-the-art results on many benchmarks. There are also deep unsupervised hashing methods proposed. For example, Deep Hashing (DH) [Erin Liong *et al.*, 2015] builds hierarchical neural networks to learn the binary codes. DeepBit [Lin *et al.*, 2016] and Unsupervised Triplet Hashing (UTH) [Huang *et al.*, 2017] learn the hashing functions with data argumentation. Binary Deep Neural Network (UH-BDNN) [Do *et al.*, 2016] was developed by maintaining a discrete binary constraint during training. DBD-MQ [Duan *et al.*, 2017] obtains the hash codes by using K-AutoEncoders network to minimize the reconstruction errors.

Different from the previously proposed deep unsupervised hashing methods, our proposed DeepQuan utilizes the product quantization method to generate binary codes. It simultaneously refines the clustering structure and preserves the data manifold. Furthermore, the proposed weighted triplet loss helps to avoid trivial solutions and poor generalization. We show that our method achieves the state-of-the-art performance on two public datasets for the image retrieval application. For more evaluation results, readers can refer to https://github.com/chenjunjie1994/IJCAI-18.

3 Methodology

In this section, we describe the proposed DeepQuan in detail. We use bold lowercase letters like $\mathbf{x}$ to indicate vectors and uppercase letters like X to denote matrices. $\|\cdot\|_2$ and $\|\cdot\|_0$ denote L2 and L0 norms respectively. Given a training set with N image feature vectors $X = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i$ denotes the D-dimensional feature vector of the i^{th} image. The unsupervised hashing problem is defined as learning a mapping function $f : \mathbf{x} \mapsto \mathbf{h} \in \{0,1\}^B$, which encodes a feature vector into a B-bit binary vector, so that the pairwise similarity in the original feature space is preserved as much as possible.

Our proposed model DeepQuan is built upon a deep autoencoder network, and the whole architecture is shown in Fig. 1. It accepts feature vectors (i.e., hand-crafted features or deep features) as inputs, and consists of three components: (1) a stack of fully-connected layers used as an en-



(a) Average Distance vs Epoch (b) Loss Function vs Epoch

Figure 2: Computing the binary codes on MNIST dataset @ 32bit. (a) The average distance between codewords which is computed using Eq. (1) and Eq. (3) keeps decreasing, resulting in trivial solutions (*i.e.* all the codewords become the identical) and thus poor generalization. It is obvious that Eq.(4) ends up with a much larger average distance, which leads to more discriminative structure. (b) The loss value decreases until convergence as the training proceeds.

coder to map high-dimensional inputs to low-dimensional compact representations; (2) a symmetric decoder for preserving the manifold structure within data; (3) a proposed weighted triplet loss on low-dimensional representations for training the deep model and learning the binary codes.

3.1 Model Formulation

The model configuration of our deep network follows the setting of [Xie *et al.*, 2016]. It is a symmetric architecture, in which both the encoder and decoder consist of five fully connected layers but reversed to each other. Each fully connected layer learns a nonlinear mapping $\mathbf{z}_i^l = \alpha^l(W^l\mathbf{z}_i^{l-1} + \mathbf{b}^l)$, where $\mathbf{z}_i^l$ is the l^{th} hidden representation of input data $\mathbf{x}_i$, W^l and $\mathbf{b}^l$ are the weight and bias parameters of the l^{th} layer respectively. The activation functions for all the layers except the last fully-connected layer are rectifier units (ReLU) $\alpha(x) = max(0, x)$. Unless otherwise specified, we use $\mathbf{z}_i$, $f(\cdot)$ and $g(\cdot)$ to represent the middle hidden representation of $\mathbf{x}_i$, the encoder and the decoder respectively in the rest of the paper.

Weighted Triplet Loss We employ product quantization (PQ) to generate binary codes from the middle hidden representation $\mathbf{z}_i \in \mathcal{R}^L$ by minimizing the quantization error. Product quantization is one special case of vector quantization (VQ), where any codeword is taken from the Cartesian product of M sub-codebooks [Ge *et al.*, 2014]. Thus, we first decompose the representation $\mathbf{z}_i$ into M parts $\mathbf{z}_i = [\mathbf{z}_{i1}, \mathbf{z}_{i2}, \ldots, \mathbf{z}_{iM}]^T$, where $\mathbf{z}_{im} \in \mathcal{R}^{L/M}$ is the subvector related to m^{th} subspace. In each subspace, K codewords can be generated using K-means clustering to quantize all the subvectors $\{\mathbf{z}_{im}\}_{i=1}^N$. Finally, there will be K^M codewords in total. The way to measure the quantization error, as adopted in [Cao *et al.*, 2016], can take a quadratic form, given as:

$$\sum_{m=1}^{M}\sum_{i=1}^{N}\|\mathbf{z}_{im} - C_m\mathbf{h}_{im}\|_2^2 \quad s.t.\ \mathbf{h}_{im} \in \{0,1\}^K, \|\mathbf{h}_{im}\|_0 = 1 \tag{1}$$

where C_m denotes the codebook associated with the m^{th} subspace. The codebook C_m is composed of K codewords $C_m = [C_{m1}, C_{m2}, \ldots, C_{mK}]$, where $C_{mk} \in \mathcal{R}^{L/M}$. The constraint indicates that $\mathbf{h}_{im}$ is a one-hot K-dimensional binary vector, which means each subvector is represented by only one codeword. Following [Ge *et al.*, 2014; Cao *et al.*, 2016; Long *et al.*, 2016], to represent each data point $\mathbf{x}_i$ with a compact binary vector, each $\mathbf{h}_{im}$ is compressed into one log_2K-bit vector, with which the final binary vector is the concatenation of M log_2K-bit vectors. In other words, the length of hash codes $B = Mlog_2K$. For all the experiments we conduct, we set $K = 256$. Each component thus is represented with 8 bits.

One can simply take Eq. (1) as the objective function and alternatively optimize it with respect to $\mathbf{z}_i$ and C_m to learn the deep model. However, the result will not be desirable. We conducted a preliminary experiment and found that all the inferred codewords within a codebook are more or less the same as indicated by the average distance among them shown in Fig. 2(a). This indicates that the hidden representations $\mathbf{z}_i$ do not exhibit an obvious clustering structure for the product quantization to leverage on. We further confirm this by visualizing the learned feature space in Fig. 3.

To promote the clustering structure to be inferred for the hidden representations, we propose a novel weighted triplet loss that takes all the codewords (i.e., clusters) into consideration instead of only the assigned codewords to learn $\mathbf{z}_i$ during the training. This is to allow more discriminative clustering structure to be obtained. In particular, the proposed weighted triplet loss is formulated as:

$$J(\mathbf{z}_i, C_i^+, C_i^-) = max\{s - (\lambda\|\mathbf{z}_i - C_i^-\|_2 - \|\mathbf{z}_i - C_i^+\|_2), 0\} \quad s.t.\ \lambda \in (0,1) \tag{2}$$

where $C_i^+ = [C_{i1}^+, C_{i2}^+ \ldots C_{iM}^+]$ and $C_i^- = [C_{i1}^-, C_{i2}^- \ldots C_{iM}^-]$ denote the concatenations of all positive and negative codewords respectively. Here, positive codewords are defined as the cluster centers to which $\mathbf{z}_{im}$ is assigned via K-means clustering. Negative codewords are defined as those cluster centers to which $\mathbf{z}_{im}$ is not assigned. In our experiments, we also define a parameter s which can be interpreted as the "margin" between the positive and the negative codewords. Negative codewords are generated by randomly sampling.

Compared with the simple quadratic loss in Eq. (1) which only considers the relation with positive codewords, our proposed weighted triplet loss simultaneously enlarges the distance between negative codewords C_i^- and $\mathbf{z}_i$ so that a discriminative clustering structure can be obtained, which will further benefit the quantization process. We add a weighting parameter $\lambda \in (0, 1)$ to the negative term in Eq. (2) to control the emphasis on learning discriminative representations $\mathbf{z}_i$.

Manifold Preserving Loss To ensure that the hidden representation $\mathbf{z}_i$ can well recover the data manifold in the original feature space , we employ a decoder to define the regularization term formulated as follows:

$$R(\mathbf{z}_i) = \|\mathbf{x}_i - g(\mathbf{z}_i)\|_2^2 \tag{3}$$

where $\mathbf{x}_i$ is the input feature vector and $g(\cdot)$ denotes the decoder network.

Overall Objective We formulate the overall objective function for model training as:

$$L_\theta = \sum_{i=1}^{N} J(\mathbf{z}_i, C_i^+, C_i^-) + \eta R(\mathbf{z}_i) \tag{4}$$

where θ denotes the parameters of the deep neural network, and η is a trade-off parameter to control the balance between feature discrimination and reconstruction quality so as to lead to better generalization.

3.2 Approximate Nearest Neighbor Search

For testing, given a new query point $\mathbf{x}_q$, we compute a symmetric distance between $\mathbf{x}_q$ and the data points $\mathbf{x}_i$ in the image database based on the inferred binary codes $\{\mathbf{h}_i\}_{i=1}^N$ and the codebooks C for the retrieval. Specifically, we adopt the one used in [Jegou *et al.*, 2011], defined as:

$$SD(x_q, x_i) = \sum_{m=1}^{M} \|q(\mathbf{x}_q)_m - C_m \mathbf{h}_{im}\|_2^2. \tag{5}$$

To compute that, we first precompute the distance look-up table of size $K \times K$ between the codewords. Thus, there will be M $K \times K$ look-up tables altogether. With the M pre-computed look-up tables, the distance $SD(x_q, x_i)$ can be efficiently computed by summing up the codeword distance values directly obtained from the look-up tables. It is only slightly more costly than computing the Hamming distance [Jegou *et al.*, 2011].

Note that the distance computed with Eq. (5) is an approximation of the Euclidean distance between $\mathbf{x}_q$ and $\mathbf{x}_i$. As discussed in [Jegou *et al.*, 2011], we show that the error is statistically upper bounded by the quantization error.

Theorem 1. The statistical error between the approximated distance $SD(\mathbf{x}_q, \mathbf{x}_i)$ and the Euclidean distance $d(\mathbf{x}_q, \mathbf{x}_i)$ is upper bounded as :

$$|SD(\mathbf{x}_q, \mathbf{x}_i) - d(\mathbf{x}_q, \mathbf{x}_i)| \leq |d(\mathbf{x}_q, q(\mathbf{x}_q)) + d(\mathbf{x}_i, q(\mathbf{x}_i))| \tag{6}$$

Proof. The upper bound can be derived by simply using the triangle inequality:

$$\begin{aligned} &|SD(\mathbf{x}_q, \mathbf{x}_i) - d(\mathbf{x}_q, \mathbf{x}_i)| \\ &\quad = |d(q(\mathbf{x}_q), q(\mathbf{x}_i)) - d(\mathbf{x}_q, \mathbf{x}_i)| \\ &\quad \leq |d(\mathbf{x}_q, q(\mathbf{x}_i)) + d(\mathbf{x}_q, q(\mathbf{x}_q)) - d(\mathbf{x}_q, \mathbf{x}_i)| \\ &\quad \leq |d(\mathbf{x}_i, q(\mathbf{x}_i)) + d(\mathbf{x}_q, q(\mathbf{x}_q))| \end{aligned}$$

3.3 Alternating Optimization

We propose to optimize Eq. (4) in an alternating way. We learn the compact representation $\mathbf{z}$, the codebooks C, and the binary codes $\mathbf{h}$ iteratively until convergence.

Learning Model Parameters θ and $\mathbf{z}$ Standard back propagation (BP) is adopted to update the model parameters θ and $\mathbf{z}$. We simply take the gradient of the objective function L_θ with respect to $\mathbf{z}$ as follows:

$$\begin{aligned} \frac{\partial L_\theta}{\partial \mathbf{z}_i} =& \frac{\partial J(\mathbf{z}_i, C_i^+, C_i^-)}{\partial \mathbf{z}_i} \cdot \mathbf{1}\{s - (\lambda \|\mathbf{z}_i - C_i^-\|_2 - \|\mathbf{z}_i - C_i^+\|_2)\} \\ &+ 2\eta(\mathbf{x}_i - g(\mathbf{z}_i)) \frac{\partial g(\mathbf{z}_i)}{\partial \mathbf{z}_i} \end{aligned} \tag{7}$$

where $\mathbf{1}\{s - (\lambda \|\mathbf{z}_i - C_i^-\|_2 - \|\mathbf{z}_i - C_i^+\|_2)\}$ is an indicator function which will return 1 if $s - (\lambda \|\mathbf{z}_i - C_i^-\|_2 - \|\mathbf{z}_i - C_i^+\|_2) > 0$, and 0 otherwise. To optimize the deep model, we back propagate the gradient $\frac{\partial \mathbf{z}_i}{\partial \theta}$ through the chain rule and update the parameters θ using the gradient descent method.

Learning C and $\mathbf{h}$ As shown in Theorem 1, the quality (i.e. measured by quantization error) of codebooks C has crucial impact on the final retrieval performance. By minimizing the quantization error, we can obtain the optimal codebooks C. For the m^{th} subspace, we have

$$\sum_{i=1}^{N} \|\mathbf{z}_{im} - C_m \mathbf{h}_{im}\|_2^2$$

and the estimated values of C and $\mathbf{h}$ are obtained using K-means following [Jegou *et al.*, 2011]. In our implementation, we update the codebook C and $\mathbf{h}$ every epoch. As Eq. (4) tries to drive the hidden features to have a clustering structure, the updating of the cookbook for each cluster will eventually be only affected by the hidden representations assigned to it, but not the others. This makes the alternating optimization to converge. Fig. 2(b) shows that the objective function Eq. (4) converges as the training proceeds.

4 Experimental Results

To evaluate the performance of our proposed DeepQuan method for image retrieval, we conduct extensive experiments on two publicly available datasets. The details of the experiments and the results are described in the following subsections.

4.1 Datasets and Experimental Setting

To make our performance comparison consistent with some recent related work [Lu *et al.*, 2017; Huang *et al.*, 2017; Erin Liong *et al.*, 2015], we here conduct our experiments on two widely used benchmark datasets: CIFAR-10 and MNIST.

CIFAR-10 The CIFAR-10 dataset consists of $60,000$ labeled color images in 10 classes. Each class contains $6,000$ images of size 32×32.

MNIST The MNIST dataset contains $70,000$ gray-scale images in 10 classes of digits from "0" to "9". Each image is of size 28×28.

To demonstrate the effectiveness of the proposed DeepQuan model, we compare it with several state-of-the-art unsupervised hashing methods including: **DH** [Lu *et al.*, 2017; Erin Liong *et al.*, 2015], **DeepBit** [Lin *et al.*, 2016], **DBD-MQ** [Duan *et al.*, 2017], **UH-BNN** [Do *et al.*, 2016], **UTH** [Huang *et al.*, 2017], **ITQ** [Gong *et al.*, 2013], **KMH** [He *et al.*, 2013], **SH** [Weiss *et al.*, 2009], **LSH** [Datar *et al.*, 2004], **AGH** [Liu *et al.*, 2011], **Spherical** [Heo *et al.*, 2012], **PCAH** [Wang *et al.*, 2012]. For the comparison on dataset MNIST, we follow the setting of **DH** [Lu *et al.*, 2017; Erin Liong *et al.*, 2015]. Specifically, we randomly sample $1,000$ images (100 images per class) to form the test set (*i.e.*, queries) and take the rest $69,000$ images as the training set and the gallery database. Each image is represented by a

Method	MNIST (MAP@All)			CIFAR-10-GIST (MAP@1000)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
ITQ	41.18	43.82	45.37	15.67	15.67	16.64
KMH	32.12	33.29	35.78	13.59	13.59	14.46
SH	25.81	30.77	24.10	12.55	12.55	12.56
Spherical	26.64	25.72	34.75	13.98	14.58	15.38
PCAH	27.33	24.85	31.71	12.91	12.60	12.10
LSH	20.88	25.83	31.71	12.55	13.76	15.07
AGH	39.92	33.39	28.64	13.64	13.61	13.54
DH	43.14	44.97	46.74	16.17	16.62	16.96
UH-BNN	45.38	47.21	–	17.83	**18.52**	–
DeepQuan	**60.30**	**55.50**	**52.54**	**18.19**	18.20	**18.74**

Table 1: Mean Average Precision (%) for different numbers of bits on two datasets. The best MAPs are shown in bold.

784-D gray-scale feature vector with the pixel values as its elements. For the experiments on the dataset CIFAR-10, we put them under two categories according to the input feature vector being used. Following the settings of DH and UH-BNN, we compare DeepQuan with those methods whose inputs are 512-D GIST [Oliva and Torralba, 2001] feature vectors extracted from the raw images, including DH, UH-BNN, ITQ, KMH, SH, LSH, AGH, Spherical and PCAH. We denote the CIFAR-10 dataset with these features as CIFAR-10-GIST. For the CIFAR-10-GIST dataset, again $1,000$ images (100 images per class) are randomly sampled as the query data. The rest $59,000$ images are used as the training set. For fair comparison with the other deep models (*i.e.*, DeepBit, DBD-MQ, UTH) with raw pixel images as the input and using convolutional neural network (CNN) for feature learning, we extract 7^{th} fully-connected layer from the deep VGG-16 [Simonyan and Zisserman, 2014] network as the feature vector inputs for our model. This VGG-16 model is also used in DBD-MQ [Duan *et al.*, 2017] as part of the model. We denote this dataset as CIFAR-10-CNN, where $10,000$ images are sampled as the query set and the rest $50,000$ images are used for training.

Following [Xie *et al.*, 2016], we configure our model as a deep autoencoder with the number of unit of encoder as $[D-500-500-2000-L]$, where D and L are the dimensions of the input vector and the middle hidden layer respectively. The dimension L is essentially the length of binary codes, specifically $L = 16M$ as [Cao *et al.*, 2016]. The trade-off parameter $\eta = 1.0$ is adopted for all the experiments. Also, we have tested the cases empirically with $s = 1.0, 1.0, 0.001$ and $\lambda = 0.1, 0.2, 0.2$ for the datasets MNIST, CIFAR-10-CNN and CIFAR-10-GIST respectively. As adopted in most of the related work, the mean average precision (MAP) is used to measure the retrieval performance. For the performance of the existing methods, we make direct reference to the results reported in the corresponding papers. For our model, we report the average performance of 10 trials. We first pre-train the basic deep autoencoder using the training set. Then, we initialize the network accordingly and then further train the model by optimizing Eq. (4) as explained in Section 3.3. We set the batch size as 512 and the learning rate as 0.01 for the stochastic gradient descent. We found that around $3,500$ iterations are enough for the model to converge.

Method	16 bits	32 bits	64 bits
DeepBit	19.43	24.86	27.73
UTH	28.66	30.66	32.41
DBD-MQ	21.53	26.50	31.85
DeepQuan	**39.95**	**41.25**	**43.26**

Table 2: Mean Average Precision (%) (i.e. MAP@1000)for CIFAR-10-CNN dataset. The best MAPs are shown in bold.

4.2 Performance Comparison

In this subsection, we present the results of performance comparison between DeepQuan and the state-of-the-art methods. The MAP results on MNIST and CIFAR-10-GIST datasets are listed in Table 1. We can find that our DeepQuan method outperforms the state-of-the-art methods by a large margin, especially under the short-bit settings. For example, for the 16-bit experiment of MNIST, DeepQuan outperforms the second best method UH-BNN by about 15%. In Table 2, we show the results evaluated on CIFAR-10-CNN. Compared with deep unsupervised CNN models which take raw pixels images as inputs, DeepQuan also achieves the best performance. Noted that the MAP value is computed on the whole database for MNIST while it is computed on the top $1,000$ returned samples for CIFAR-10-GIST and CIFAR-10-CNN (as what being adopted in [Erin Liong *et al.*, 2015; Duan *et al.*, 2017]), which are denoted by MAP@All and MAP@1000 respectively. We believe that these large-margin improvements are due to the capability of the product quantization design incorporated into the deep model to preserve well the underlying data manifold.

4.3 Empirical Analysis of DeepQuan

To better understand how different parts of the proposed DeepQuan contribute to the overall improved performance, we conduct additional experiments. We make use of the datasets MNIST and CIFAR-10-CNN for this set of experiments, and $1,000$ images are sampled from each dataset to form a query set with the rest for training.

To evaluate the effectiveness of the proposed unified framework to learn both discriminative and compact hidden repre-

Method	MNIST		CIFAR-10-CNN	
	8 bits	16 bits	8 bits	16 bits
PQ	52.98	48.30	28.77	31.37
DAE-PQ	67.52	54.16	30.25	26.22
DeepQuan	**74.32**	**60.30**	**32.49**	**32.01**

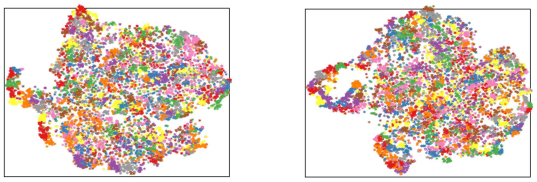
Table 3: We compare DeepQuan with two variant methods PQ and DAE-PQ to show the encoder can learn more discriminative compact features. Results are compared with MAP@All under relatively short bit setting.

Method	MNIST		CIFAR-10-CNN	
	16 bits	64 bits	16 bits	64 bits
Quad	53.18	28.12	37.53	38.49
WTPs	59.90	49.79	39.49	41.62
DeepQuan	**60.30**	**52.54**	**39.95**	**44.22**

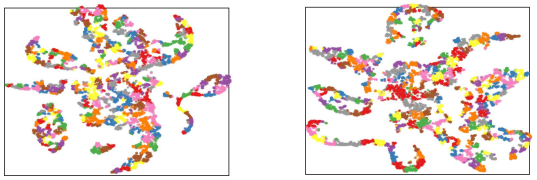
Table 4: We compare DeepQuan with two variants Quad and WTPs to show effectiveness of the proposed weighted triplet loss and inclusion of the reconstruction term. MAP@All and MAP@1000 are computed for MNIST and CIFAR-10-CNN respectively.

sentations and product quantization in DeepQuan, we conduct experiments with relatively shorter bit-length (8 bits and 16 bits), and compute the hidden representations and the product quantization differently. In particular, we run product quantization [Jegou *et al.*, 2011] directly on the original feature vectors, and on the feature vectors extracted from the pre-trained deep autoencoder, denoted as PQ and DAE-PQ respectively. The results are reported in Table 3. DeepQuan achieves the best performance. In other words, it learns more discriminative compact features which in turn can ensure more effective quantization.

Next, to evaluate the effectiveness of the proposed weighted triplet loss (Eq. (2)) as compared with the standard quadratic loss (Eq. (1)), we conduct experiments by trying variants of the overall objective function. We tried the following settings: (i) We drop the reconstruction term (i.e. $\eta = 0$) and train the model by minimizing only Eq. (1) (denoted as Quad); (ii) We drop the reconstruction term and train the model by minimizing Eq. (2) (denoted as WTPs). As shown in Table 4, Quad gives the worst performance that is anticipated, and then followed by WTPs. DeepQuan considering all the reconstruction term during the training achieves the best results. We further demonstrate that the use of the weighted triplet loss can end up with a much more discriminative clustering structure. Fig. 3 visualizes the hidden features learned from MNIST under 32-bit setting .For 32-bit setting, the hidden representation $\mathbf{z}_i$ is decomposed into 4 subspaces, each of which is expected to learn $K = 256$ clusters. Figs. 3(a) and 3(b) show the first two feature spaces learned using the quadratic loss with Eq. (3) as the manifold preserving term denoted as Quad-R, while Figs. 3(c) and 3(d) show the first two feature spaces learned with DeepQuan. It can be seen that the features learned with Quad-R are mixed, but the features learned with DeepQuan exhibit a discriminative clustering structure which benefits the subsegment quantization process.

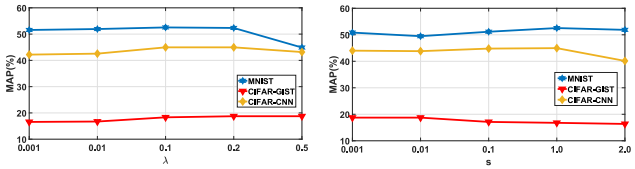


(a) Quad-R: 1^{st} subspace (b) Quad-R: 2^{nd} subspace



(c) DeepQuan: 1^{st} subspace (d) DeepQuan: 2^{nd} subspace

Figure 3: Visualization of the first two subspaces learned with Quad-R and DeepQuan on MNIST @32 bits, which is depicted with t-SNE. Best view in color.



(a) MAP@64 bits vs λ (b) MAP@64bits vs s

Figure 4: Parameters sensitivity analysis.

4.4 Parameters Sensitivity Analysis

Fig. 3 shows the effect of changing the values of the hyper-parameters s and λ. The experiments are conducted for varying one parameter at a time while fixing the other. All the results reported are under the 64 bits setting. We can find that DeepQuan can achieve competitive results within the range of $0.001 \leq \lambda \leq 0.2$ and $0.001 \leq s \leq 1.0$ compared with the existing unsupervised hashing methods.

5 Conclusions

In this paper, we propose a novel deep unsupervised method DeepQuan to learn binary codes. In contrast with existing unsupervised hashing methods, the intrinsic data manifold can be better preserved through the integration of product quantization and deep models in an unified framework. Experimental results on two widely used datasets show that our method outperforms the existing state-of-the-art models for image retrieval. An empirical analysis is also conducted to illustrate the effectiveness of the proposed weighted triplet loss in achieving discriminative hidden representations. For future work, we will explore the model with convolution neural networks where features can be directly learned from the raw images to further improve the retrieval accuracy.

References

[Andoni and Indyk, 2006] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[Cao *et al.*, 2016] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262. ACM, 2004.

[Do *et al.*, 2016] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, pages 219–234. Springer, 2016.

[Duan *et al.*, 2017] Yueqi Duan, Jiwen Lu, Ziwei Wang, Jianjiang Feng, and Jie Zhou. Learning deep binary descriptor with multi-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1183–1192, 2017.

[Erin Liong *et al.*, 2015] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

[Ge *et al.*, 2014] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014.

[Gong *et al.*, 2013] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[Gray, 1984] Robert Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984.

[He *et al.*, 2013] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2938–2945, 2013.

[Heo *et al.*, 2012] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[Huang *et al.*, 2017] Shanshan Huang, Yichao Xiong, Ya Zhang, and Jia Wang. Unsupervised triplet hashing for fast image retrieval. *arXiv preprint arXiv:1702.08798*, 2017.

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

[Li *et al.*, 2015] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[Lin *et al.*, 2016] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[Liu *et al.*, 2011] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1–8. Citeseer, 2011.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE, 2012.

[Long *et al.*, 2016] Mingsheng Long, Yue Cao, Jianmin Wang, and Philip S Yu. Composite correlation quantization for efficient multimodal retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 579–588. ACM, 2016.

[Lu *et al.*, 2017] Jiwen Lu, Venice Erin Liong, and Jie Zhou. Deep hashing for scalable image search. *IEEE Transactions on Image Processing*, 26(5):2352–2367, 2017.

[Oliva and Torralba, 2001] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Wang *et al.*, 2012] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Processings of International Conference on Machine Learning*, pages 478–487, 2016.

[Zhang *et al.*, 2017] Jian Zhang, Yuxin Peng, and Mingkuan Yuan. Unsupervised generative adversarial cross-modal hashing. *arXiv preprint arXiv:1712.00358*, 2017.