Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Deep Quantization Network for Efficient Image Retrieval*

Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu and **Qingfu Wen**
School of Software, Tsinghua University, Beijing, China
Tsinghua National Laboratory for Information Science and Technology
{caoyue10,zhuhan10,qingfu.wen}@gmail.com {mingsheng,jimwang}@tsinghua.edu.cn

Abstract

Hashing has been widely applied to approximate nearest neighbor search for large-scale multimedia retrieval. Supervised hashing improves the quality of hash coding by exploiting the semantic similarity on data pairs and has received increasing attention recently. For most existing supervised hashing methods for image retrieval, an image is first represented as a vector of hand-crafted or machine-learned features, then quantized by a separate quantization step that generates binary codes. However, suboptimal hash coding may be produced, since the quantization error is not statistically minimized and the feature representation is not optimally compatible with the hash coding. In this paper, we propose a novel Deep Quantization Network (DQN) architecture for supervised hashing, which learns image representation for hash coding and formally control the quantization error. The DQN model constitutes four key components: (1) a sub-network with multiple convolution-pooling layers to capture deep image representations; (2) a fully connected bottleneck layer to generate dimension-reduced representation optimal for hash coding; (3) a pairwise cosine loss layer for similarity-preserving learning; and (4) a product quantization loss for controlling hashing quality and the quantizability of bottleneck representation. Extensive experiments on standard image retrieval datasets show the proposed DQN model yields substantial boosts over latest state-of-the-art hashing methods.

Introduction

While image big data with large volume and high dimension are pervasive in search engines and social networks, it has attracted increasing attention to enable approximate nearest neighbors (ANN) retrieval of images with both computation efficiency and search quality. An advantageous solution is hashing methods (Wang et al. 2014), which transform high-dimensional data into compact binary codes and generate similar binary codes for similar data items. In this paper, we focus on learning to hash methods that build data-dependent hash coding for efficient image retrieval, which have shown better performance than data-independent hashing methods, e.g. Locality-Sensitive Hashing (LSH) (Gionis et al. 1999).

*Corresponding authors: Mingsheng Long and Jianmin Wang.

Many learning to hash methods have been proposed to enable efficient ANN search using Hamming distance (Kulis and Darrell 2009; Gong and Lazebnik 2011; Norouzi and Blei 2011; Fleet, Punjani, and Norouzi 2012; Liu et al. 2012; Wang, Kumar, and Chang 2012; Liu et al. 2013; Gong et al. 2013; Yu et al. 2014; Xia et al. 2014; Zhang et al. 2014; Shen et al. 2015; Lai et al. 2015; Erin Liong et al. 2015). Hash learning can be divided into unsupervised methods and supervised methods. While unsupervised methods are more general and can be trained without semantic labels or relevances, they are restricted by the semantic gap dilemma (Smeulders et al. 2000) that high-level semantic description of an object often differs from low-level feature descriptors. Supervised methods can incorporate semantic labels or relevances to mitigate the semantic gap and improve the hashing quality, i.e. achieve accurate search with fewer bits of codes.

Recently, deep learning to hash methods (Xia et al. 2014; Lai et al. 2015) have shown that both feature representation and hash coding can be learned more effectively using deep neural networks (Krizhevsky, Sutskever, and Hinton 2012; Lin, Chen, and Yan 2014), which can naturally encode any nonlinear hashing functions. These deep hashing methods have created state-of-the-art results on many benchmarks. However, a crucial disadvantage of these deep learning to hash methods is that the quantization error is not statistically minimized and the feature representation is not optimally compatible with binary hash coding. Note that, not all input vectors can be quantized effectively using vector quantization (VQ)—if input vectors do not exhibit a cluster structure, then they may not be quantized accurately (Ge et al. 2014). Hence it is important to improve the quantizability of the deep image representations such that they can be quantized more effectively. Another limitation is that these methods do not adopt a well-specified pairwise loss to link the pairwise distances with the similarity labels, i.e. to classify whether a data pair is similar or dissimilar (pairwise classification) based on the pairwise distances. Therefore, suboptimal hash coding may be produced by existing deep hashing methods.

In this paper, we put forward a novel Deep Quantization Network (DQN) architecture for supervised hashing, which learns deep image representation compatible for hash coding and formally controls the quantization error in an optimization framework. The DQN architecture constitutes four key components: (1) a sub-network with multiple convolution-

pooling layers to capture good image representations; (2) a fully connected bottleneck layer to generate dimension-reduced representation that is optimal for hash coding; (3) a pairwise cosine loss layer for similarity-preserving learning; and (4) a product quantization loss for controlling hashing quality and the quantizability of bottleneck representation. Extensive experiments on standard image retrieval datasets show that the proposed DQN architecture yields substantial improvements over current state-of-the-art hashing methods.

Related Work

Existing learning to hash methods can be categorized in two categories: unsupervised hashing and supervised hashing. We refer readers to (Wang et al. 2014) for a recent survey.

Unsupervised hashing methods learn hash functions that can encode input data points to binary codes only using the unlabeled training data. Typical learning criteria include reconstruction error minimization (Salakhutdinov and Hinton 2007; Jegou, Douze, and Schmid 2011), neighborhood preserving as graph-based hashing (Weiss, Torralba, and Fergus 2009; Liu et al. 2011), and quantization error minimization as Iterative Quantization (ITQ) (Gong and Lazebnik 2011).

Supervised hashing explores supervised information (e.g., class labels, relative similarity, or relevance feedback) to learn compact hash coding. Binary Reconstruction Embedding (BRE) (Kulis and Darrell 2009) pursues hash functions by minimizing the squared errors between the distances of data points and the distances of corresponding hash codes. Minimal Loss Hashing (MLH) (Norouzi and Blei 2011) and Hamming Distance Metric Learning (Norouzi, Blei, and Salakhutdinov 2012) learn hash codes by minimizing hinge-like loss functions based on relative similarity of data points. Supervised Hashing with Kernels (KSH) (Liu et al. 2012) is a kernel-based method that builds compact binary codes by minimizing the Hamming distances on similar pairs and maximizing the Hamming distances on dissimilar pairs.

Recent revolution in deep learning shows that deep convolutional neural network (CNN) (Krizhevsky, Sutskever, and Hinton 2012; Lin, Chen, and Yan 2014; Bengio, Courville, and Vincent 2013) can automatically learn effective image representations that yield breakthrough performance on general computer vision tasks. Xia et al. proposed CNNH (Xia et al. 2014) that decomposes the hash learning process into a stage of learning approximate hash codes, followed by a deep-network-based stage of simultaneously fine-tuning the image features and hash functions. Lai et al. improved the two-stage CNNH by proposing DNNH (Lai et al. 2015), a simultaneous feature learning and hash coding deep network such that image representations and hash codes can improve each other in the joint learning process. DNNH has created the latest state-of-the-art results on many benchmarks.

This work further improves DNNH by exploiting three key problems: (1) control the quantization error in a principled way, (2) devise a pairwise cosine loss to better link the pairwise cosine distances with the similarity labels, and (3) learn the similarity-preserving deep representation that is optimal for hash coding. The three improvements constitute the proposed Deep Quantization Network (DQN) approach.

Deep Quantization Network

In similarity retrieval, we are given a training set of N points $\{\boldsymbol{x}_i\}_{i=1}^N$, each represented as D-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^D$. Some pairs of points are associated with similarity labels s_{ij}, where $s_{ij} = 1$ implies $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar and $s_{ij} = -1$ indicates $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. Our goal is to learn nonlinear hashing function $f : \boldsymbol{x} \mapsto \boldsymbol{h} \in \{-1, 1\}^B$ to encode each point $\boldsymbol{x}$ in compact B-bit hash code $\boldsymbol{h} = f(\boldsymbol{x})$ such that the similarity between given pairs is preserved. In supervised hashing, $\mathcal{S} = \{s_{ij}\}$ is usually constructed from the semantic labels within the data points or the relevance feedback from click-through data in image retrieval systems.

In this paper, we propose a Deep Quantization Network (DQN) architecture for hash learning, as shown in Figure 1. This architecture accepts input images in a pairwise form $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$ and processes them through the deep representation learning and hash coding pipeline: (1) a sub-network with multiple convolution-pooling layers to extract good image representations; (2) a fully-connected bottleneck layer to generate optimal dimension-reduced representation; (3) a pairwise cosine loss layer for similarity-preserving learning; and (4) a product quantization loss for controlling hashing quality and the quantizability of bottleneck representation.

Model Formulation

We start with AlexNet (Krizhevsky, Sutskever, and Hinton 2012), the deep convolutional neural network (CNN) comprised of five convolutional layers ($conv1$–$conv5$) and three fully connected layers ($fc6$–$fc8$). Each fc layer ℓ learns a nonlinear mapping $\boldsymbol{z}_i^\ell = a^\ell\left(\boldsymbol{W}^\ell \boldsymbol{z}_i^{\ell-1} + \boldsymbol{b}^\ell\right)$, where $\boldsymbol{z}_i^\ell$ is the ℓ-th layer hidden representation of point $\boldsymbol{x}_i$, $\boldsymbol{W}^\ell$ and $\boldsymbol{b}^\ell$ are the weight and bias parameters of the ℓ-th layer, and a^ℓ is the activation function, taken as rectifier units (ReLU) $a^\ell(\boldsymbol{x}) = \max(\boldsymbol{0}, \boldsymbol{x})$ for all hidden layers $conv1$–$fc7$. For hash learning, we replace the $fc8$ layer of the softmax classifier in the original AlexNet with a new bottleneck layer fcb of R units, which transforms the $fc7$ layer representation to R-dimensional bottleneck representation $\boldsymbol{z}_i^l$, where $l = 8$ is the total number of layers. To encourage the fcb layer representation $\boldsymbol{z}_i^l$ to be optimal for hash coding, we use the hyperbolic tangent (tanh) activation function $a^l(\boldsymbol{x}) = \tanh(\boldsymbol{x})$ to produce nonlinear dimension-reduced representation.

In this paper, we guarantee that the fcb representation $\boldsymbol{z}_i^l$ will be optimal for hash coding by jointly (1) preserving the similarity between given pairs in $\mathcal{S}$, (2) controlling the quantization error of binarizing the fcb representation $\boldsymbol{z}_i^l$ into binary codes $\boldsymbol{h}_i$, and (3) improving the quantizability of the fcb representation $\boldsymbol{z}_i^l$ so that it can be quantized effectively.

Pairwise Cosine Loss For a pair of binary codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, there is a nice relationship between their Hamming distance $\mathrm{dist}_H(\cdot,\cdot)$ and inner product $\langle\cdot,\cdot\rangle$: $\mathrm{dist}_H\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \frac{1}{2}\left(B - \langle \boldsymbol{h}_i, \boldsymbol{h}_j\rangle\right)$. Hence, we can use the inner product as a surrogate of the Hamming distance to quantify the pairwise similarity. However, as our goal is to learn the optimal fcb representation $\boldsymbol{z}_i^l$ for hash coding while $\boldsymbol{z}_i^l$ is continuous, the range of inner product $\left\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \right\rangle \in [-R, R]$ is not consistent with the binary labels $s_{ij} \in \{-1, 1\}$. Therefore, we propose

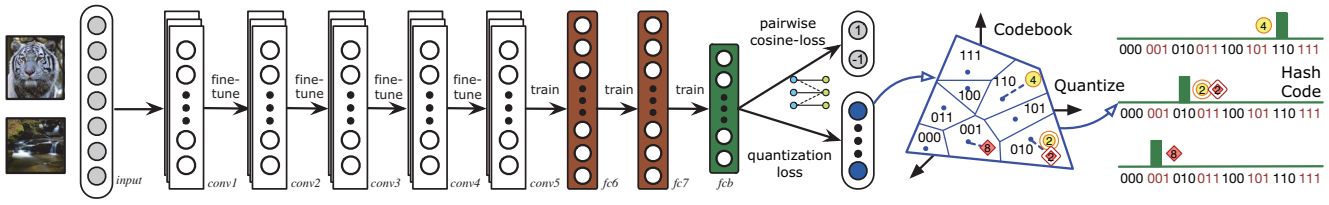


Figure 1: Deep Quantization Network (DQN) with multiple convolution-pooling layers $conv1$–$fc7$ for representation learning, a fully-connected bottleneck layer fcb for optimal dimensionality reduction, a pairwise cosine loss for similarity-preserving learning, and a product quantization loss for controlling the hashing quality and the quantizability of bottleneck representation.

a novel pairwise squared loss using the cosine distance to quantify the similarity between pairs of fcb representations

$$L = \sum_{s_{ij} \in \mathcal{S}} \left(s_{ij} - \frac{\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \rangle}{\|\boldsymbol{z}_i^l\| \, \|\boldsymbol{z}_j^l\|} \right)^2, \tag{1}$$

where $\|\cdot\|$ is the vector length. The range of cosine distance $\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \rangle / \|\boldsymbol{z}_i^l\| \, \|\boldsymbol{z}_j^l\| \in [-1, 1]$ is consistent with the binary similarity labels $s_{ij} \in \{-1, 1\}$, hence making Equation (1) a well-specified loss for preserving the pairwise similarity information conveyed in $\mathcal{S}$. In real-world retrieval systems, cosine distance is widely adopted to mitigate the diversity of vector lengths and improve the retrieval quality, while it has not been well explored for supervised hash learning (Wang et al. 2014). It is worth noting that, the proposed pairwise cosine loss (1) is better-specified than the widely-used pairwise inner-product loss $L = \sum_{s_{ij} \in \mathcal{S}} \left(s_{ij} - \frac{1}{B} \langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \rangle \right)^2$ (Liu et al. 2012; Xia et al. 2014), because $\frac{1}{B} \langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \rangle \in [-1, 1]$ will not hold for representation $\{\boldsymbol{z}_i^l\}$ with continuous relaxation.

Product Quantization Loss We propose to employ the state-of-the-art product quantization (PQ) (Ge et al. 2014) approach to construct compact binary hash code $\boldsymbol{h}_i$ from the similarity-preserving bottleneck representation $\boldsymbol{z}_i^l$. PQ is a solution to Vector Quantization (VQ) when an exponentially large number of codewords are desired to accurately reconstruct the input vectors. The key idea is to decompose the original vector space into the Cartesian product of M low-dimensional subspaces and quantize each subspace into K codewords (clusters) via K-means clustering. Specifically, we partition the bottleneck representation into M subspaces, i.e. $\boldsymbol{z}_i^l = [\boldsymbol{z}_{i1}^l; \ldots; \boldsymbol{z}_{iM}^l]$, where $\boldsymbol{z}_{im}^l \in \mathbb{R}^{R/M}$ is the sub-vector of $\boldsymbol{z}_i^l$ associated with the m-th subspace. Then we quantize all sub-vectors $\{\boldsymbol{z}_{im}^l\}_{i=1}^N$ of each subspace m into K clusters (codewords) independently through K-means as

$$\begin{aligned} Q &= \sum_{m=1}^{M} \sum_{i=1}^{N} \left\| \boldsymbol{z}_{im}^l - \boldsymbol{C}_m \boldsymbol{h}_{im} \right\|_2^2 \\ \|\boldsymbol{h}_{im}\|_0 &= 1, \boldsymbol{h}_{im} \in \{0, 1\}^K, \end{aligned} \tag{2}$$

where $\boldsymbol{C}_m = [\boldsymbol{c}_{m1}, \ldots, \boldsymbol{c}_{mK}] \in \mathbb{R}^{\frac{R}{M} \times K}$ denotes the codebook of K codewords (cluster centers) in the m-th subspace, and $\boldsymbol{h}_{im}$ is the one-of-K encoding indicating which one (and only one) of the K codewords in the m-th codebook $\boldsymbol{C}_m$ is selected to approximate the i-th point $\boldsymbol{z}_i^l$. Denote by $\boldsymbol{h}_i = [\boldsymbol{h}_{i1}; \ldots; \boldsymbol{h}_{iM}] \in \mathbb{R}^{MK}$ the encoding of point $\boldsymbol{z}_i^l$ that concatenates all M subspace one-of-K encodings $\{\boldsymbol{h}_{im}\}$. Since each $\boldsymbol{h}_{im}$ can be compressed in $\log_2 K$ bits, the final hash codes $\boldsymbol{h}_i$ can be compacted in $B = M \log_2 K$ bits.

To guarantee that the fcb representation $\boldsymbol{z}_i^l$ will be optimal for hash coding, we jointly (1) control the quantization error of binarizing the fcb representation $\boldsymbol{z}_i^l$ into binary codes $\boldsymbol{h}_i$, and (2) improve the quantizability of the fcb representation $\boldsymbol{z}_i^l$ so that it can be quantized effectively. In this regard, we can rewrite Equation (2) in compact matrix form as follows

$$Q = \sum_{i=1}^{N} \left\| \boldsymbol{z}_i^l - \boldsymbol{C} \boldsymbol{h}_i \right\|_2^2, \tag{3}$$

where codebook $\boldsymbol{C} \in \mathbb{R}^{R \times MK}$ is a block diagonal matrix

$$\boldsymbol{C} = \operatorname{diag}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_M) = \begin{bmatrix} \boldsymbol{C}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{C}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{C}_M \end{bmatrix}. \tag{4}$$

By minimizing Equation (3), we can control the quantization error of converting the continuous bottleneck representation $\boldsymbol{z}_i^l$ into compact binary code $\boldsymbol{h}_i$. Moreover, we can improve the quantizability of the bottleneck representation $\boldsymbol{z}_i^l$ such that it can be quantized more effectively. As shown in (Ge et al. 2014), not all input vectors can be quantized effectively using the product quantization (PQ)—if input vectors do not exhibit a cluster structure, then they may not be quantized accurately, as is a common sense for data clustering. In this paper, we propose to update $\{\boldsymbol{z}_i^l\}$ by Equation (3) such that they can be reconstructed more accurately with codebook $\boldsymbol{C}$ and binary code $\{\boldsymbol{h}_i\}$. Hence, we can make the bottleneck representation well quantizable and optimal for hash coding.

Optimization Problem In this paper, we perform simultaneous representation learning and hash coding by integrating Equation (1) and (3) into a unified optimization problem as

$$\min_{\Theta, \boldsymbol{C}, \boldsymbol{H}} L + \lambda Q, \tag{5}$$

where $\lambda > 0$ is trade-off parameter between pairwise cosine loss L and product quantization loss Q, and $\Theta \triangleq \{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\}$ denotes the set of network parameters. Through joint optimization problem (5), we can achieve statistically optimal

learning of the hash codes, by jointly preserving the pairwise similarity in training data and controlling the quantization error of binarizing continuous embeddings to binary codes.

Approximate Nearest Neighbor Search Approximate nearest neighbor (ANN) search with Euclidean distance is a powerful task for quantization techniques. Given a database of DQN hash codes $\{\boldsymbol{h}_i\}_{i=1}^N$, we follow (Jegou, Douze, and Schmid 2011) and use the *Asymmetric Quantizer Distance* (AQD) as similarity metric, which computes the Euclidean distance between query $\boldsymbol{q}$ and database point $\boldsymbol{x}_i$ as follows,

$$\text{AQD}\left(\boldsymbol{q}, \boldsymbol{x}_i\right) = \sum_{m=1}^{M} \left\| \boldsymbol{z}_{qm}^l - \boldsymbol{C}_m \boldsymbol{h}_{im} \right\|_2^2, \tag{6}$$

where $\boldsymbol{z}_q^l$ is the bottleneck representation of query $\boldsymbol{q}$, and $\boldsymbol{h}_{im}$ is the binary code of point $\boldsymbol{x}_i$ associated with the m-th codebook $\boldsymbol{C}_m$. For computation speedup, for each query $\boldsymbol{q}$, the Euclidean distances between $\boldsymbol{q}$ and all the codewords in M codebooks can be pre-computed and stored in a query-specific $M \times K$ lookup table, which is used to compute AQD between the query and all database points $\boldsymbol{h}_i$, each entails M table lookups and additions and is only slightly more costly than Hamming distance (Jegou, Douze, and Schmid 2011). We can also build the inverted multi-indexing (Babenko and Lempitsky 2015) of the DQN hash codes for non-exhaustive search, but will not elaborate it here due to space limitation.

Learning Algorithm

Learning Θ We derive the learning algorithm for the DQN model (5), and show rigorously that both pairwise cosine loss and product quantization loss can be optimized efficiently through the standard back-propagation (BP) procedure. For notation brevity, we denote the point-wise cost as

$$\begin{aligned} C_i &= L_i + \lambda Q_i \\ &= \sum_{j:s_{ij}\in\mathcal{S}} \left(s_{ij} - \frac{\left\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \right\rangle}{\left\| \boldsymbol{z}_i^l \right\| \left\| \boldsymbol{z}_j^l \right\|} \right)^2 + \lambda \left\| \boldsymbol{z}_i^l - \boldsymbol{C}\boldsymbol{h}_i \right\|_2^2. \end{aligned} \tag{7}$$

Then we derive the gradient of point-wise cost C_i w.r.t. $\boldsymbol{W}_k^\ell$, the network parameter of the k-th unit in the ℓ-th layer as

$$\begin{aligned} \frac{\partial C_i}{\partial \boldsymbol{W}_k^\ell} &= \frac{\partial L_i}{\partial \boldsymbol{W}_k^\ell} + \lambda \frac{\partial Q_i}{\partial \boldsymbol{W}_k^\ell} \\ &= \left(\frac{\partial L_i}{\partial \hat{z}_{ik}^\ell} + \lambda \frac{\partial Q_i}{\partial \hat{z}_{ik}^\ell} \right) \frac{\partial \hat{z}_{ik}^\ell}{\partial \boldsymbol{W}_k^\ell} \\ &= \delta_{ik}^\ell \boldsymbol{z}_i^{\ell-1}, \end{aligned} \tag{8}$$

where $\hat{\boldsymbol{z}}_i^\ell = \boldsymbol{W}^\ell \boldsymbol{z}_i^{\ell-1} + \boldsymbol{b}^\ell$ is the output of the ℓ-th layer before activation function $a^\ell(\cdot)$, and $\delta_{ik}^\ell \triangleq \frac{\partial L_i}{\partial \hat{z}_{ik}^\ell} + \lambda \frac{\partial Q_i}{\partial \hat{z}_{ik}^\ell}$ is the point-wise *residual* term that measures how much the k-th unit in the ℓ-th layer is responsible for the error of point $\boldsymbol{x}_i$ in the network output. For an output unit k, we can directly measure the difference between the network's activation and the true target value, and use it to define the residual δ_{ik}^l as

$$\begin{aligned} \delta_{ik}^l = &\sum_{j:s_{ij}\in\mathcal{S}} \left[\left(\frac{\left\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \right\rangle}{\left\| \boldsymbol{z}_i^l \right\| \left\| \boldsymbol{z}_j^l \right\|} - s_{ij} \right) z_{jk}^l \right] \dot{a}^l\left(\hat{z}_{ik}^l\right) \\ &+ \lambda \left(z_{ik}^l - \boldsymbol{c}_{k*} \boldsymbol{h}_i \right) \dot{a}^l\left(\hat{z}_{ik}^l\right), \end{aligned} \tag{9}$$

where $l = 8$ denotes the index of the output layer, $\dot{a}^l(\cdot)$ is the derivative of the l-th layer activation function, and $\boldsymbol{c}_{k*}$ is the k-th row of codebook matrix $\boldsymbol{C}$ in Equation (4). For a hidden unit k in the $(\ell-1)$-th layer, we compute the residual $\delta_{ik}^{\ell-1}$ based on a weighted average of the errors of all the units $k' = 1, \ldots, u_\ell$ in the ℓ-th layer that involve $\boldsymbol{z}_i^{\ell-1}$ as an input, which is just consistent with standard BP procedure,

$$\delta_{ik}^{\ell-1} = \left(\sum_{k'=1}^{u_\ell} \delta_{ik'}^\ell W_{k'k}^\ell \right) \dot{a}^{\ell-1}\left(\hat{z}_{ik}^{\ell-1}\right), \tag{10}$$

where u_ℓ is the number of hidden units in the ℓ-th layer. The residuals in all layers can be computed by back-propagation.

An important property of the proposed algorithm is that, only computing the residual of the output layer involves the pairwise summation as in Equation (9). For all hidden layers, all the residuals can be simply computed recursively by Equation (10), which does not involve pairwise summation. Hence we do not need to modify the implementation of BP in all hidden layers $1 \leq \ell \leq l-1$. We only need modify standard BP by replacing the output residual with Equation (9).

Since the only difference between standard BP and our algorithm is Equation (9), we analyze the computational complexity based on Equation (9). Denote the number of similarity pairs $\mathcal{S}$ available for training as $|\mathcal{S}|$, then it is easy to verify that the computational complexity is linear $O(|\mathcal{S}|)$.

Learning $\boldsymbol{C}$ and $\boldsymbol{H}$ Given bottleneck representation $\{\boldsymbol{z}_i^l\}$ fixed, codebook $\boldsymbol{C} = \text{diag}(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_M)$ and binary codes $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$ can be learned through M independent K-means by optimizing Equation (2). Specifically, for each subspace m, we perform the K-means algorithm as follows

$$\min_{\boldsymbol{C}_m, \boldsymbol{h}_{im} \in \{0,1\}^K} \sum_{i=1}^{N} \left\| \boldsymbol{z}_{im}^l - \boldsymbol{C}_m \boldsymbol{h}_{im} \right\|_2^2. \tag{11}$$

This product quantization can be computed for each epoch, with linear-time sample complexity $O(N)$ (Ge et al. 2014).

Theoretical Analysis

Given a query $\boldsymbol{q}$ and a database point $\boldsymbol{x}_i$ with hash code $\boldsymbol{h}_i$, their Euclidean distance can be computed as $d\left(\boldsymbol{q}, \boldsymbol{x}_i\right) = \left\| \boldsymbol{z}_q^l - \boldsymbol{z}_i^l \right\|_2$, where $\boldsymbol{z}_q^l$ and $\boldsymbol{z}_i^l$ are the deep representations of $\boldsymbol{q}$ and $\boldsymbol{x}_i$ respectively. The motivation of hashing is that computing the Euclidean distance directly on real-valued vectors is too costly for large-scale image retrieval. Hence, we compute AQD (6) between query $\boldsymbol{q}$ and binary code $\boldsymbol{x}_i$ to approximate the Euclidean distance, and we need to analyze approximation error. Denote $\hat{\boldsymbol{z}}_i^l = \boldsymbol{C}\boldsymbol{h}_i$ the reconstruction of $\boldsymbol{z}_i^l$, then $\text{AQD}\left(\boldsymbol{q}, \boldsymbol{x}_i\right) = d\left(\boldsymbol{q}, \hat{\boldsymbol{x}}_i\right) + \epsilon$ where ϵ is a constant.

Theorem 1 (Error Bound)**.** *The error of using AQD* (6) *to approximate original Euclidean distance is bounded by* (3)

$$\left| \text{AQD}\left(\boldsymbol{q}, \boldsymbol{x}_i\right) - d\left(\boldsymbol{q}, \boldsymbol{x}_i\right) \right| \leqslant \left\| \boldsymbol{z}_i^l - \boldsymbol{C}\boldsymbol{h}_i \right\|_2 + |\epsilon|. \tag{12}$$

Proof. From the triangle inequality, it follows that

$$\begin{aligned}|\mathrm{AQD}(\boldsymbol{q}, \boldsymbol{x}_i) - d(\boldsymbol{q}, \boldsymbol{x}_i)| &= |(d(\boldsymbol{q}, \hat{\boldsymbol{x}}_i) + \epsilon) - d(\boldsymbol{q}, \boldsymbol{x}_i)| \\ &\leqslant |d(\boldsymbol{q}, \hat{\boldsymbol{x}}_i) - d(\boldsymbol{q}, \boldsymbol{x}_i)| + |\epsilon| \\ &\leqslant d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) + |\epsilon| \\ &= \left\|\boldsymbol{z}_i^l - \boldsymbol{C}\boldsymbol{h}_i\right\|_2 + |\epsilon|.\end{aligned}$$

□

The above theorem confirms that the error of using AQD to approximate the Euclidean distance on real-valued vectors is statistically bounded by DQN quantization loss (3). Thus DQN is more accurate than sign thresholding methods that do not control the quantization error (Lai et al. 2015).

Experiments

We conduct extensive experiments to evaluate the efficacy of the proposed DQN model against several state-of-the-art hashing methods on three widely-used benchmark datasets. The codes and configurations will be made available online.

Evaluation Setup

We conduct extensive empirical evaluation on three public benchmark datasets, **NUS-WIDE**, **CIFAR-10**, and **Flickr**.

- **NUS-WIDE**[1] is a public web image dataset. We follow the settings in (Liu et al. 2011; Lai et al. 2015) and use the subset of 195,834 images that are associated with the 21 most frequent concepts, where each concept consists of at least 5,000 images. We resize all images into 256×256.
- **CIFAR-10**[2] is a dataset containing 60,000 color images in 10 classes, and each class has 6,000 images in size 32×32.
- **Flickr**[3] consists of 25,000 images collected from Flickr, where each image is labeled with one of the 38 semantic concepts. We resize images of this subset into 256×256.

We follow the experimental protocols in (Lai et al. 2015). In NUS-WIDE and CIFAR-10, we randomly select 100 images per class as the test query set, and 500 images per class as the training set. In Flickr, we randomly select 1000 images as the test query set, and 4000 images as the training set. The similarity pairs for training are randomly constructed using image labels: each pair is considered similar (dissimilar) if they share at least one (none) semantic label.

We follow (Lai et al. 2015) to evaluate the image retrieval quality based on three widely-adopted evaluation metrics: Mean Average Precision (MAP) for different numbers of bits, Precision-Recall curves, and Precision curves with respect to different numbers of top returned samples. For fair comparison, all methods use identical training and test sets.

We evaluate and compare the retrieval performance of the proposed **DQN** approach and its variants with nine state-of-the-art hashing methods, including three unsupervised methods **LSH** (Gionis et al. 1999), **SH** (Weiss, Torralba, and Fergus 2009) and **ITQ** (Gong and Lazebnik 2011), and seven supervised methods **DNNH** (Lai et al. 2015), **CNNH** (Xia et al. 2014), **KSH** (Liu et al. 2012), **MLH** (Norouzi and Blei 2011), **BRE** (Kulis and Darrell 2009) and **ITQ-CCA** (Gong and Lazebnik 2011). To highlight the advantage of DQN, we also report the results of **KSH-D**, the best shallow baseline KSH using DeCAF$_7$ features (Donahue et al. 2014) extracted from the AlexNet model pre-trained on ImageNet.

For the deep learning based methods, including CNNH, DNNH and DQN, we directly use the raw image pixels as the input. For the shallow learning based methods, we follow (Liu et al. 2012; Lai et al. 2015; Srivastava and Salakhutdinov 2014) to represent each image in NUS-WIDE by a 500-dimensional bag-of-words vector, to represent each image in CIFAR-10 by a 512-dimensional GIST vector, and to represent each image in Flickr by a 3,857-dimensional vector concatenated by local SIFT feature, global GIST feature, etc. All image features are available at the datasets' website.

To guarantee that our results directly comparable to most published results, the results of LSH, BRE, ITQ, ITQ-CCA, KSH, MLH and SH on both the NUS-WIDE and CIFAR-10 datasets are directly reported from the latest work (Lai et al. 2015), while the results on the Flickr dataset are obtained by the implementations provided by their authors, following standard cross-validation procedures for model selection.

We implement the DQN model based on the open-source **Caffe** framework (Jia et al. 2014). We employ the AlexNet architecture (Krizhevsky, Sutskever, and Hinton 2012), fine-tune convolutional layers $conv1$–$conv5$ and fully-connected layers $fc6$–$fc7$ that were copied from the pre-trained model, and train hashing layer fch, all via back-propagation. As the fch layer is trained from scratch, we set its learning rate to be 10 times that of the lower layers. We use the mini-batch stochastic gradient descent (SGD) with 0.9 momentum and the learning rate annealing strategy implemented in Caffe, and cross-validate learning rate from 10^{-5} to 10^{-2} with a multiplicative step-size 10. We also fix the mini-batch size of images as 64 and the weight decay parameter as 0.0005.

For the product quantization loss, we cross-validate the λ from 10^{-5} to 1 with a multiplicative step-size 10. We follow (Ge et al. 2014) and adopt $K = 256$ codewords for each codebook in each subspace. For each data point, the binary hash code associated with the Cartesian product of all the M subspaces requires $B = M \log_2 K = 8M$ bits (i.e. M bytes) for compact hash coding, where we set $M = B/8$. We follow similar strategy in (Lai et al. 2015) and set the bottleneck dimension $R = 16M$ such that the product quantizer can quantize the bottleneck representations accurately.

Results and Discussions

The MAP results of all methods are listed in Table 1, which show the proposed DQN method substantially outperforms all the comparison methods. Specifically, compared to the best baseline using traditional hand-crafted visual features, KSH, we achieve absolute increases of **20.5%**, **22.9%** and **15.1%** in average MAP for different bits on NUS-WIDE, CIFAR-10, and Flickr respectively. It is desirable that DQN significantly outperforms KSH-D (the best shallow baseline KSH using DeCAF$_7$ features) by **7.5%**, **2.5%** and **6.4%** increments in average MAP for three benchmark datasets.

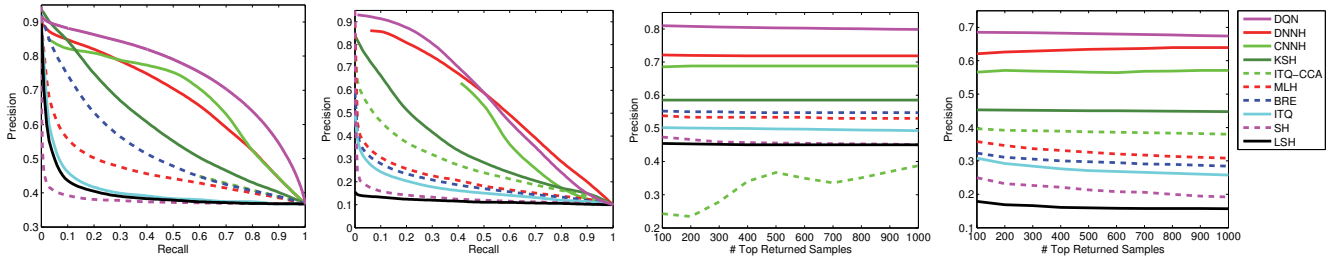
[1] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

[2] http://www.cs.toronto.edu/kriz/cifar.html

[3] http://press.liacs.nl/Flickr/

Table 1: Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on Three Image Datasets

Method	NUS-WIDE				CIFAR-10				Flickr			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
SH	0.433	0.426	0.426	0.423	0.131	0.135	0.133	0.130	0.531	0.533	0.531	0.529
LSH	0.403	0.421	0.426	0.441	0.121	0.126	0.120	0.120	0.499	0.513	0.521	0.548
ITQ	0.452	0.468	0.472	0.477	0.162	0.169	0.172	0.175	0.544	0.555	0.560	0.570
ITQ-CCA	0.435	0.435	0.435	0.435	0.264	0.282	0.288	0.295	0.513	0.531	0.540	0.555
MLH	0.500	0.514	0.520	0.522	0.182	0.195	0.207	0.211	0.610	0.618	0.629	0.634
BRE	0.485	0.525	0.530	0.544	0.159	0.181	0.193	0.196	0.571	0.592	0.599	0.604
KSH	0.556	0.572	0.581	0.588	0.303	0.337	0.346	0.356	0.690	0.702	0.702	0.706
KSH-D	0.673	0.705	0.717	0.725	0.502	0.534	0.558	0.563	0.777	0.786	0.792	0.793
CNNH	0.617	0.663	0.657	0.688	0.484	0.476	0.472	0.489	0.749	0.761	0.768	0.776
DNNH	0.674	0.697	0.713	0.715	0.552	**0.566**	0.558	**0.581**	0.783	0.789	0.791	0.802
DQN	**0.768**	**0.776**	**0.783**	**0.792**	**0.554**	0.558	**0.564**	0.580	**0.839**	**0.848**	**0.854**	**0.863**



(a) NUS-WIDE (b) CIFAR-10 (c) NUS-WIDE (d) CIFAR-10

Figure 2: The results of comparison methods on the NUS-WIDE and CIFAR-10 datasets: (a)-(b) precision-recall curves @ 48 bits; (c)-(d) precision w.r.t. top returned samples curves @ 48 bits. DQN is superior for precision-oriented retrieval systems.

Compared to the state-of-the-art deep-network-based hashing method, DNNH, the proposed DQN approach outperforms it by very large margins of **8.0%** and **5.9%** in average MAP for different bits on the NUS-WIDE and Flickr datasets respectively, and achieve comparable MAP results on the CIFAR-10 dataset. It is somewhat unexpected that KSH-D even significantly outperforms CNNH and achieves comparable results with DNNH. This highlights the importance of designing well-specified loss functions for training deep hashing networks. DNNH uses a fixed-margin loss and piecewise-linear activation function to train deep networks, which may cause information loss and objective oscillations in back propagation. In the proposed DQN approach, (1) we design a novel pairwise cosine loss for training CNN such that the pairwise distances can be better linked with the similarity labels; and (2) we further update the bottleneck representation by minimizing the PQ loss to improve the quantizability of the bottleneck representation. Both improvements of DQN contribute significantly to its superior performance.

Figures 2 shows the precision-recall curves and precision w.r.t. top returned samples curves @ 48 bits on NUS-WIDE and CIFAR-10, respectively. From the curves, we can observe that DQN outperforms all the comparison methods by very large margins, including the latest state-of-the-art method DNNH. It is also worth noting that, although DQN obtains comparable MAP results with DNNH on CIFAR-10, DQN significantly outperforms DNNH in the two curves on the CIFAR-10 dataset, demonstrating that DQN can be more favorable for the precision-oriented image retrieval systems.

Empirical Analysis

To evaluate the effectiveness of the pairwise cosine loss for similarity-preserving learning and the product quantization loss for controlling hash quality, we design three variants of the proposed DQN approach: (1) a two-step method $\mathbf{DQN}_2$, which separately learns the bottleneck representations via CNN and the compositional binary codes via PQ (Jegou, Douze, and Schmid 2011); (2) $\mathbf{DQN}_{2+}$, an enhanced variant of DQN_2 using optimized PQ (OPQ) (Ge et al. 2014); (3) $\mathbf{DQN}_{\mathbf{ip}}$, a DQN variant that utilizes the widely-adopted pairwise inner-product loss $L = \sum_{s_{ij} \in \mathcal{S}} \left(s_{ij} - \frac{1}{B} \left\langle \boldsymbol{z}_i^l, \boldsymbol{z}_j^l \right\rangle\right)^2$ (Liu et al. 2012; Xia et al. 2014) instead of the proposed pairwise cosine loss (1). The MAP results w.r.t. different numbers of bits on the three datasets are reported in Table 2.

We can observe that, by simultaneously preserving similarity information using pairwise cosine loss (1) and controlling hashing quality using product quantization loss (3), DQN outperforms DQN_2 and DQN_{2+} by 1.8%, 2.6%, 3.3% and 2.4%, 2.9%, 3.7% respectively in average MAP. Furthermore, DQN_2 using PQ performs even slightly better than DQN_{2+} with OPQ, testifying that the quantizability of deep representations cannot be simply improved by optimizing shallow quantization models. It is indispensable to improve the quantizability of deep representations by optimizing the product quantization loss when training the deep networks.

Table 2: Mean Average Precision (MAP) Results of DQN and Its Variants, DQN_2, DQN_{2+}, and DQN_{ip} on Three Datasets

Method	NUS-WIDE				CIFAR-10				Flickr			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
DQN_2	0.755	0.763	0.764	0.766	0.533	0.537	0.542	0.545	0.806	0.815	0.821	0.831
DQN_{2+}	0.750	0.754	0.756	0.764	0.528	0.534	0.538	0.541	0.804	0.809	0.815	0.829
DQN_{ip}	0.623	0.646	0.655	0.673	0.506	0.513	0.519	0.529	0.748	0.756	0.759	0.775
DQN	**0.768**	**0.776**	**0.783**	**0.792**	**0.554**	**0.558**	**0.564**	**0.580**	**0.839**	**0.848**	**0.854**	**0.863**

Another crucial observation is that, by using the pairwise cosine loss (1), DQN can outperform DQN_{ip} using the pairwise inner-product loss by very large margins of 9.1%, 4.7% and 13.1% in average MAP. The pairwise inner-product loss has been widely adopted in previous work (Liu et al. 2012; Xia et al. 2014). However, this loss does not link well the pairwise distances between points (taking values in $(-R, R)$ when using continuous relaxation) to the pairwise similarity labels (taking binary values $\{$-1,1$\}$). In contrast, the pairwise cosine loss is inherently consistent with the training pairs.

Conclusion

In this paper, we have formally approached the problem of supervised deep hashing in a joint optimization framework. The proposed Deep Quantization Network (DQN) architecture simultaneously optimizes the pairwise cosine loss on semantic similarity pairs and the product quantization loss on compact hash codes. Extensive experiments on standard image retrieval datasets show that the DQN architecture yields substantial boosts over the state-of-the-art hashing methods.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61502265), National Natural Science Funds for Distinguished Young Scholars (No. 613250154), China Postdoctoral Science Foundation (No. 2015T80088), and Tsinghua National Laboratory (TNList) Special Funds for Big Data Science and Technology.

References

Babenko, A., and Lempitsky, V. 2015. The inverted multi-index. *TPAMI* 37(6):1247–1260.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *TPAMI* 35:1798–1828.

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Erin Liong, V.; Lu, J.; Wang, G.; Moulin, P.; and Zhou, J. 2015. Deep hashing for compact binary codes learning. In *CVPR*.

Fleet, D. J.; Punjani, A.; and Norouzi, M. 2012. Fast search in hamming space with multi-index hashing. In *CVPR*. IEEE.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2014. Optimized product quantization. *TPAMI*.

Gionis, A.; Indyk, P.; Motwani, R.; et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*, 518–529. ACM.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*.

Gong, Y.; Kumar, S.; Rowley, H.; Lazebnik, S.; et al. 2013. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 484–491. IEEE.

Jegou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *TPAMI* 33(1):117–128.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*. ACM.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

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

Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*.

Lin, M.; Chen, Q.; and Yan, S. 2014. Network in network. In *ICLR, 2014 (arXiv:1409.1556)*.

Liu, W.; Wang, J.; Kumar, S.; and Chang, S.-F. 2011. Hashing with graphs. In *ICML*. ACM.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; and Chang, S.-F. 2012. Supervised hashing with kernels. In *CVPR*. IEEE.

Liu, X.; He, J.; Lang, B.; and Chang, S.-F. 2013. Hash bit selection: a unified solution for selection problems in hashing. In *CVPR*.

Norouzi, M., and Blei, D. M. 2011. Minimal loss hashing for compact binary codes. In *ICML*, 353–360. ACM.

Norouzi, M.; Blei, D. M.; and Salakhutdinov, R. R. 2012. Hamming distance metric learning. In *NIPS*, 1061–1069.

Salakhutdinov, R., and Hinton, G. E. 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, 412–419.

Shen, F.; Shen, C.; Liu, W.; and Tao Shen, H. 2015. Supervised discrete hashing. In *CVPR*. IEEE.

Smeulders, A. W.; Worring, M.; Santini, S.; Gupta, A.; and Jain, R. 2000. Content-based image retrieval at the end of the early years. *TPAMI* 22(12):1349–1380.

Srivastava, N., and Salakhutdinov, R. 2014. Multimodal learning with deep boltzmann machines. *JMLR* 15:2949–2980.

Wang, J.; Shen, H. T.; Song, J.; and Ji, J. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*.

Wang, J.; Kumar, S.; and Chang, S.-F. 2012. Semi-supervised hashing for large-scale search. *TPAMI* 34(12):2393–2406.

Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *NIPS*.

Xia, R.; Pan, Y.; Lai, H.; Liu, C.; and Yan, S. 2014. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2156–2162. AAAI.

Yu, F. X.; Kumar, S.; Gong, Y.; and Chang, S.-F. 2014. Circulant binary embedding. In *ICML*, 353–360. ACM.

Zhang, P.; Zhang, W.; Li, W.-J.; and Guo, M. 2014. Supervised hashing with latent factor models. In *SIGIR*, 173–182. ACM.