Push for quantization
Deep fisher hashing

Li, Yunqiang; Pei, Wenjie; Zha, Yufei; Van Gemert, Jan

Push for Quantization: Deep Fisher Hashing

Yunqiang Li*[1]
y.li-19@tudelft.nl
Wenjie Pei*[2]
wenjiecoder@outlook.com
Yufei zha*[3]
zhayufei@126.com
Jan van Gemert[1]
j.c.vangemert@tudelft.nl

[1] Vision Lab, Delft University of Technology, Netherlands
[2] Tencent, China
[3] School of Computer Science, Northwestern Polytechnical University, Xi'an, China

Abstract

Current massive datasets demand light-weight access for analysis. Discrete hashing methods are thus beneficial because they map high-dimensional data to compact binary codes that are efficient to store and process, while preserving semantic similarity. To optimize powerful deep learning methods for image hashing, gradient-based methods are required. Binary codes, however, are discrete and thus have no continuous derivatives. Relaxing the problem by solving it in a continuous space and then quantizing the solution is not guaranteed to yield separable binary codes. The quantization needs to be included in the optimization. In this paper we push for quantization: We optimize maximum class separability in the binary space. We introduce a margin on distances between dissimilar image pairs as measured in the binary space. In addition to pair-wise distances, we draw inspiration from Fisher's Linear Discriminant Analysis (Fisher LDA) to maximize the binary distances between classes and at the same time minimize the binary distance of images within the same class. Experiments on CIFAR-10, NUS-WIDE and ImageNet100 demonstrate compact codes comparing favorably to the current state of the art.

1 Introduction

Image hashing aims to map high-dimensional images onto compact binary codes where pair-wise distances between binary codes corresponds to semantic image distances, *i.e.,* Similar binary codes should have similar class labels. Binary codes are efficient to store and have low computational cost which is particularly relevant in today's big data age where huge datasets demand fast processing.

A problem in applying powerful deep learning methods for image hashing is that deep nets are optimized using gradient descent while binary codes are discrete and thus have no continuous derivatives and cannot be directly optimized by gradient descent. The current solution [2, 12, 20, 22, 37, 39] is to relax the discrete problem to a continuous one, and after optimization in the continuous space, quantize it to obtain discrete codes. This approach, however, disregards the importance of the quantization, which is problematic because image

* Both authors contributed equally. * Corresponding Author

class similarity in the continuous space is not necessarily preserved in the binary space, as illustrated in Fig. 1. The quantization needs to be included in the optimization.

In this paper we go beyond preserving semantic distances in the continuous space: We push for quantization by optimizing maximum class separability in the binary space. To do so, we introduce a margin on distances between dissimilar image pairs explicitly measured in the binary space. In addition to pair-wise distances, we draw inspiration from Fisher's Linear Discriminant Analysis (Fisher LDA) to maximize the binary distances between classes and at the same time minimize the binary distance of images within the same class

We have the following contributions. 1) Adding a margin to pairwise labels pushes dissimilar samples apart in the binary space; 2) Fisher's criterion to maximize the between-class distance and to minimize the within-class distance leads to compact hash codes; 3) We show how to optimize this under discrete constraints and 4) We outperform state-of-the-art methods on two datasets, being particular advantageous for a small number of hashing bits.

2 Related work

Amount of supervision. Existing hashing methods can be grouped on the amount of prior domain knowledge. Hashing methods without prior knowledge are applicable to any domain and include well-known methods such as Locality-Sensitive Hashing (LSH) [7] and its extensions [5, 15, 16, 26, 28]. If some knowledge about the data distribution is known in the form of an unlabeled training set, this knowledge can be advantageously exploited by unsupervised methods [8, 10, 11, 13, 24, 25, 33] which learn hash functions by preserving the training set distance distribution. With the availability of additional prior knowledge about how samples should be grouped together, supervised methods [4, 9, 21, 27, 29, 30, 36] can leverage such label information. Particularly successful supervised hashing methods use deep learning [18, 22, 23, 34, 35] to learn the feature representation. Supervision can be in the form of pairwise label information [2, 3, 19, 20, 39] or in the form of class labels [9, 19, 23, 30]. In this paper we exploit both pairwise and class label knowledge, leading to highly compact and discriminative hash codes.

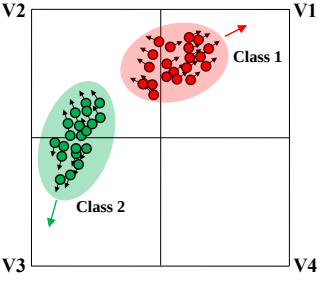


Figure 1: Example of two separable classes in a continuous space. After quantization (assign to grid cells) the classes are no longer separable. In this paper we aim for separability in the binary space.

Quantization in hashing. Several methods optimize the continue space and apply the *sign* to obtain binary codes [2, 4, 12, 20, 22, 24, 37, 38, 39]. A quantization loss is proposed in deep learning based hashing [2, 12, 20, 22, 38, 39] to force the learned continuous representations to approach the desired binary codes. However, optimizing quantization alone may not preserve class separability in the binary space. An elegant solution is to employ *sigmoid* or *tanh* to approximate the non-smooth *sign* function [3, 17], but unfortunately comes with the drawback that such activation functions have difficulty to converge when using gradient descent methods. We circumvent these limitations by imposing the quantization loss in the discrete space, optimizing the separability in the hashing space directly while guiding parameter optimization in the continuous space.

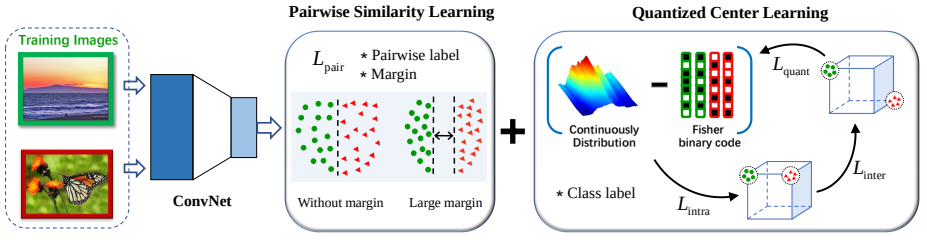


Figure 2: Images with class labels (red and green) are input to a CNN which outputs a k-dimensional continues representation $\mathbf{U}$. Module 1 maximizes a margin between dissimilar images in binary space (L_{Pair}). Module 2 minimizes binary distances within the same class (L_{Intra}) and pushes different classes away (L_{Inter}) while quantizing $\mathbf{U}$ as binary codes (L_{Quant}).

Discrete optimization. Another branch of hashing methods to solve the discrete optimization is to utilize the class information to directly learn the hashing codes. For instance, SDH [30], as well as its extensions such as FSDH [9] and DSDH [19], propose to regress the same-class images to the same binary codes. While this kind of methods encourages a close binary distance between samples from the same class, they cannot guarantee the separability of samples from different classes. In contrast, we propose to explicitly maximize the binary distances between classes and at the same time minimize the binary distances within the same class.

3 Deep Fisher Hashing with Pairwise Margin

In Fig. 2 we illustrate our model. Two components steer the discrete optimization: 1) A Pairwise Similarity Learning module to preserve semantic similarity between image pairs while using a margin to push similar and non-similar images further apart (L_{pair}). 2) A Quantized Center Learning module inspired by Fisher's linear discriminant that maximizes the distance between different-class images (L_{inter}) whilst minimizing the distance between same-class images (L_{intra}) where the binarization requires minimizing quantization errors L_{quant}. These two modules are optimized jointly on top of a convolutional network (CNN).

For a train set of N images $\mathbf{X} = \{\mathbf{x_i}\}_{i=1}^{N}$, with M class labels $\mathbf{Y} = \{\mathbf{y}_i\}_{i=n}^{N} \in \mathbb{R}^{M\times N}$, where $\mathbf{y}_i \in \mathbb{R}^M$ is a vector with all elements ≥ 0 that sums to 1, representing the class proportion of sample $\mathbf{x_i}$. For single-label (multi-class) $\mathbf{y}_i$ reverts to a one-hot encoding $\{0,1\}^M$. If $\mathbf{x}_i$ has m multiple labels, each has a value of $1/m$ in $\mathbf{y}_i$. The last layer of the CNN $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^{N} \in \mathbb{R}^{K\times N}$ is the learned representations of $\mathbf{X}$. The output codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N} \in \{-1,1\}^{K\times N}$ are the discretized binary values corresponding to $\mathbf{U}$ with each image encoded by K binary bits.

3.1 Pairwise Similarity Learning

The main goal of hashing is to have small distances between similar image pairs and large distances between dissimilar image pairs in the binary representation. For binary vectors $\mathbf{b_i}, \mathbf{b_j} \in \{-1,1\}^K$, the Hamming distance $D_H(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{2}(K - \mathbf{b}_i^\intercal \cdot \mathbf{b}_j) = \frac{1}{4} D_E(\mathbf{b}_i, \mathbf{b}_j)$. Since K is a constant, it can be left out and we define the dissimilarity $D(\mathbf{b}_i, \mathbf{b}_j) = -\frac{1}{2}\left(\mathbf{b}_i^\intercal \cdot \mathbf{b}_j\right)$. Note that larger dissimilarity D indicates larger Hamming distance and less similarity.

Similar images should share many binary values while dissimilar images should share few binary values. Given the dissimilarity $D(\cdot,\cdot) \in (-\frac{1}{2}K, \frac{1}{2}K)$, a dissimilarity of 0 between binary vectors $\mathbf{b}_i$ and $\mathbf{b}_j$ means that half of their bits are different. To encourage more overlapping bits for similar images and less overlapping bits for dissimilar images, we add a margin m to a symmetric logistic loss centered at 0:

$$L^S(D) = \log(1+e^{D+m}); L^D(D) = \log(1+e^{-D+m}). \quad (1)$$

The hyper-parameter $m \geqslant 0$ controls separation between similar pairs S and dissimilar pairs D. When $m = 0$, our model will turn into the classical way used in [19, 20]. Fig. 3 illustrates the loss curves of same-class pairs and different-class pairs as a function of dissimilarity calculated by our dissimilarity measure with various values of m. Larger margin can help to pull same-class pairs together while push different-class pairs far away.

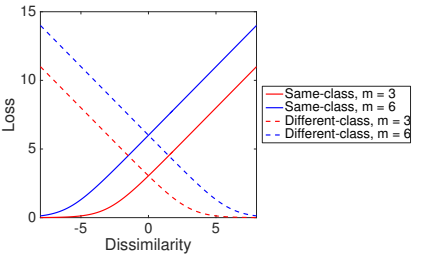


Figure 3: Our symmetric large margin logistic loss of both same-class and different-class cases as a function of the dissimilarity with different margin m. Larger m encourages separation.

The Pairwise Similarity module minimizes the large margin logistic loss:

$$\begin{aligned} L_{\text{pair}} = &\sum_{(i,j)\in\mathcal{S}} L^S(D(\mathbf{b}_i,\mathbf{b}_j)) + \sum_{(i,j)\in\mathcal{D}} L^D(D(\mathbf{b}_i,\mathbf{b}_j)) \\ &s.t. \quad \mathbf{b}_i,\mathbf{b}_j \in \{-1,1\}^K, \quad i,j = 1,...,N. \end{aligned} \quad (2)$$

Since $\mathbf{b}_i$ and $\mathbf{b}_j$ are discretized hashing codes from the continuous output of the CNN ($\mathbf{u}_i$ and $\mathbf{u}_j$), thus it is hard to back-propagate gradients from L_{pair} to parameters of the CNN. To make the CNN trainable with L_{pair}, we introduce an auxiliary variable $\mathbf{u}_i = \mathbf{b}_i$. Then we apply Lagrange multipliers to get the Lagrangian:

$$\begin{aligned} \tilde{L}_{\text{pair}} = &\sum_{(i,j)\in\mathcal{S}} L^S(D(\mathbf{u}_i,\mathbf{u}_j)) + \sum_{(i,j)\in\mathcal{D}} L^D(D(\mathbf{u}_i,\mathbf{u}_j)) + \psi \sum_{i=1}^{N} \|\mathbf{u}_i - \mathbf{b}_i\|_2^2, \\ &s.t. \quad \mathbf{b}_i,\mathbf{b}_j \in \{-1,1\}^K, \quad i,j = 1,...,N, \end{aligned} \quad (3)$$

where ψ is the Lagrange multiplier. The term $\sum_{i=1}^{N} \|\mathbf{u}_i - \mathbf{b}_i\|_2^2$ can be viewed as a constraint to minimize the discrepancy between the binary space and the continuous space.

3.2 Quantized Center Learning

The Quantized Center Learning module, see Fig. 4, maximizes the inter-class distances whilst minimizing the intra-class distances in a quantized setting. To represent class-distances we learn a center for each of the M classes: $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^{M} \in \{-1,1\}^{K\times M}$, where each center $\mathbf{c}$ is encoded by K bits of binary codes. Let $\mathbf{u}$ be the network output representation. We then encourage the learned binary code(vertex) of each representation to be close to the corresponding class center while the distance between different class centers is maximized, taking quantization to binary vectors into account.

Minimizing intra-class distances (L_{intra}). This minimizes the sum of Euclidean distance between the binary codes $\mathbf{b}_i$ of the N training images to their class center:

$$L_{\text{intra}} = \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{C}\mathbf{y}_i\|_2^2, \tag{4}$$

where all class centers $\mathbf{C}$ are indexed by $\mathbf{b}_i$'s class membership vector $\mathbf{y}_i$.

Maximizing inter-class distances (L_{inter}). We maximize the sum of pairwise Euclidean distance between different class centers to maximize the inter-class distance of training data:

$$\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \|\mathbf{c}_i - \mathbf{c}_j\|_2^2 = \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} (2K - 2\mathbf{c}_i^\intercal \mathbf{c}_j). \tag{5}$$

Since $\mathbf{c}_i, \mathbf{c}_j \in \{-1,1\}^K$ and $\mathbf{c}_i^\intercal \mathbf{c}_{j\neq i} \geq -K$, maximizing Eq. (5) is equivalent to minimizing

$$\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} (\mathbf{c}_i^\intercal \mathbf{c}_j - (-K))^2 = \|\mathbf{C}^\intercal\mathbf{C} - K(2I - J_K)\|_F^2, \tag{6}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, I is the identity matrix and J_K is the all-ones matrix. Simplifying the notation where A replaces $K(2I - J_K)$ yields

$$L_{\text{inter}} = \|\mathbf{C}^\intercal\mathbf{C} - A\|_F^2. \tag{7}$$

Minimizing quantization cost (L_{quant}). The Center Learning module exploits label information to learn binary codes by minimizing L_{intra} and L_{inter} simultaneously. We also need to encourage the learned representation to be close to the quantized binary codes. L_{quant} minimizes the total quantization cost in moving representations $\mathbf{u}_i$ towards the desired $\mathbf{b}_i$,

$$L_{\text{quant}} = \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2. \tag{8}$$

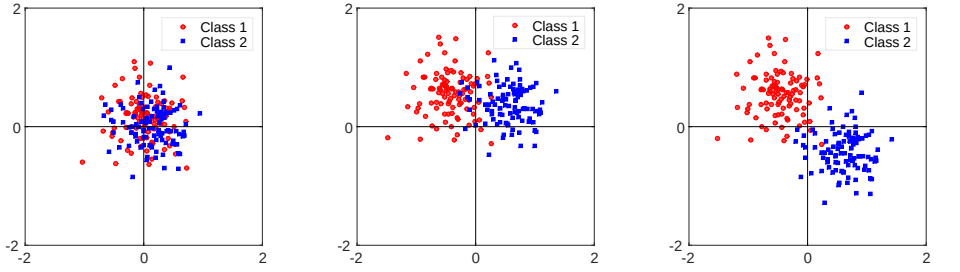
4 Optimization

Our proposed Pairwise Similarity module and Quantized Center Learning module are optimized jointly in an alternating fashion where their gradients are back-propagated to train the upstream CNN. Combining the loss functions $\tilde{L}_{pair}$ in Eq. (3), L_{intra} in Eq. (4), L_{inter} in Eq. (7) and L_{quant} in Eq. (8), the optimization of the whole framework is

$$\begin{aligned} \min_{\mathbf{b}_i,\mathbf{u}_i,\mathbf{C}} \Big[& \varphi\Big(\sum_{(i,j)\in\mathcal{S}} L^S(D(\mathbf{u}_i,\mathbf{u}_j)) + \sum_{(i,j)\in\mathcal{D}} L^D(D(\mathbf{u}_i,\mathbf{u}_j)) \Big) \\ & + \mu \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{C}\mathbf{y}_i\|_2^2 + \nu\|\mathbf{C}^\intercal\mathbf{C} - A\|_F^2 + \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2 \Big], \\ \text{s.t. } & \mathbf{C} \in \{-1,1\}^{K\times M},\ \ \mathbf{b}_i \in \{-1,1\}^K,\ \ i = 1,2,\ldots,N, \end{aligned} \tag{9}$$

where φ, μ and ν are hyper-parameters that balance the effect of three objective functions.

Optimizing Eq. (9) involves the interaction of two types of variables: discrete variables $\{\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N, \mathbf{C}\}$ and continuous variables $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^N$. A typical solution to such multi-variable optimization problem is to alternate between two steps. In particular: 1) optimize $\mathbf{U}$ while fixing $\mathbf{B}$ and $\mathbf{C}$ focusing on L_{pair} in the Pairwise Similarity Learning module, 2) fixing $\mathbf{U}$ and optimize discrete variables $\mathbf{B}$ and $\mathbf{C}$ in the Quantized Center Learning.



(a): Input (b): Only L_{intra} (c): $L_{\text{intra}} + L_{\text{inter}}$

Figure 4: Illustration of Quantized Center learning. All points denote 2D representations extracted by a CNN model from randomly selected two classes samples of CIFAR-10, for 100 samples per class. Binarization is illustrated by quantization $sgn(\cdot)$ (black lines). *(a):* Inefficient hashing: Binarization will assign same-class points to different bins, while assigning different-class points to the same bins. *(b):* Using L_{intra} clusters classes together and hashing is improved since binarization will assign the classes to different, neighboring bins: class 1 to $[-1,1]$ and class 2 to $[1,1]$. *(c):* Using $L_{\text{intra}} + L_{\text{inter}}$ also pushes the classes away from each other, improving the hashing further since after binarization class 1 is $[-1,1]$ and class 2 is $[1,-1]$ making the difference between class samples two bit flips.

4.1 Optimizing Pairwise Similarity Learning

Given $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N$, it is straightforward to optimize $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^N$ by minimizing the sub-problem resolved from Eq. (9) corresponding to L_{pair} by gradient descent:

$$\min_{\mathbf{U}} \sum_{i=1}^{m} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2 + \varphi\Big(\sum_{(i,j)\in\mathcal{S}} L^S(D(\mathbf{u}_i,\mathbf{u}_j)) + \sum_{(i,j)\in\mathcal{D}} L^D(D(\mathbf{u}_i,\mathbf{u}_j))\Big) \tag{10}$$

Since $\mathbf{U}$ is the output of the last layer of the upstream CNN, which is denoted as $\mathbf{u}_i = \mathbf{W}^\intercal \mathcal{F}_{\text{CNNs}}(\mathbf{x}_i;\theta) + \mathbf{v}$. Here $\mathbf{W}$ is the transformation matrix of the last fully connected layer and $\mathbf{v}$ is the bias term. θ is the parameters of CNNs before the last layer. For simplicity, we denote all parameters of CNNs models as $\Theta = \{\mathbf{W},\mathbf{v},\Theta\}$. The CNN parameters are optimized by gradient back-propagation: $\frac{\partial L}{\partial \Theta} = \frac{\partial L}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \Theta}$, where L is the Loss function corresponding to Eq. (10).

4.2 Optimizing Quantized Center Learning

With fixed CNN parameters Θ, we learn $\mathbf{B}$ and $\mathbf{C}$ by optimizing the Quantized Center Learning module, as:

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{C}} \mu \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{C}\mathbf{y}_i\|_2^2 + \nu\|\mathbf{C}^\intercal\mathbf{C} - A\|_F^2 + \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2, \\ &\text{s.t. } \mathbf{C} \in \{-1,1\}^{K\times M},\ \mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N \in \{-1,1\}^{K\times N}. \end{aligned} \tag{11}$$

We solve this problem by calling alternating optimization strategy again: optimize variables $\mathbf{B}$ and $\mathbf{C}$ by updating one variable with the other fixed.

Initialization of $\mathbf{b}_i$ and $\mathbf{C}$. Given the representations $\mathbf{u}_i$, we initialize $\mathbf{b}_i$ as $\mathbf{b}_i = \text{sgn}(\mathbf{u}_i)$. In the first iteration we initialize the class centers $\mathbf{C}$ with the class mean of the output representations, later we update $\mathbf{C}$ directly.

Fix $\mathbf{b}_i$, update $\mathbf{C}$. Keeping $\mathbf{b}_i$ fixed in Eq. (11) reduces this sub-problem to

$$\begin{aligned} &\min_{\mathbf{C}} \mu \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{C}\mathbf{y}_i\|_2^2 + \nu \|\mathbf{C}^\top \mathbf{C} - A\|_F^2, \\ &\text{s.t. } \mathbf{C} \in \{-1,1\}^{K\times M}. \end{aligned} \tag{12}$$

Due to the discrete constraints on the class centers $\mathbf{C}$, the minimization of above problem is a discrete optimization problem which is hard to optimize directly. We introduce an auxiliary variable $\mathbf{V}$ with the constrain $\mathbf{C} = \mathbf{V}$, and adding the Lagrange multiplier, the optimization of Eq. (12) is:

$$\begin{aligned} &\min_{\mathbf{C},\mathbf{V}} \mu \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{V}\mathbf{y}_i\|_2^2 + \nu \|\mathbf{V}^\top \mathbf{V} - A\|_F^2 + \eta \|\mathbf{C} - \mathbf{V}\|_F^2, \\ &\text{s.t.} \quad \mathbf{C} \in \{-1,1\}^{K\times M}. \end{aligned} \tag{13}$$

Fixing $\mathbf{V}$, since the optimal solution for $\mathbf{C}$ for minimizing $\|\mathbf{C} - \mathbf{V}\|_F^2$ is $\mathbf{C} = \mathrm{sgn}(\mathbf{V})$, hence $\|\mathbf{C} - \mathbf{V}\|_F^2$ in Eq. (13) can be replaced with $\|\mathrm{sgn}(\mathbf{V}) - \mathbf{V}\|_F^2$. Let $\mathcal{L}_2$ denote the loss function after applying Lagrange multipliers, then the gradient w.r.t. $\mathbf{V}$ is calculated as:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{V}} = 2\mu(\mathbf{V}\mathbf{Y} - \mathbf{B})\mathbf{Y}^\top + 4\nu\mathbf{V}(\mathbf{V}^\top\mathbf{V} - A) + 2\eta(\mathbf{V} - \mathrm{sgn}(\mathbf{V})), \tag{14}$$

approximating the class center $\mathbf{C}$ with the learned $\mathbf{V}$.

Fix C, update $\mathbf{b}_i$. With the variable $\mathbf{C}$ fixed in Eq. (11), we optimize the binary code $\mathbf{b}_i$ with the sub-problem

$$\begin{aligned} &\min_{\mathbf{b}_i} \mu \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{C}\mathbf{y}_i\|_2^2 + \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2, \\ &\text{s.t. } \mathbf{b}_i \in \{-1,1\}^K, i = 1,\ldots,N. \end{aligned} \tag{15}$$

We have the closed-form solution of problem (15):

$$\mathbf{B} = \mathrm{sgn}(\mu\mathbf{C}\mathbf{Y} + \mathbf{U}). \tag{16}$$

See the supplementary for the detailed proof. By defining $\mathcal{F} = \mu\mathbf{C}\mathbf{Y} + \mathbf{U}$ as the Fisher's transformed representations, we note that $\mathcal{F}$ is a *translation* transformation of original representations $\mathbf{U}$ which pushes different-class points to different vertex and pulls same-class points to same vertex, while $\mathcal{F}$ does not change the relative position between same class. The learned center $\mathbf{C}$ determines where the corresponding class translates to. The 2D example in Fig. 4 shows that the shape within a class does not change, yet the classes do translate.

4.3 Joint Optimization

We update the two modules jointly, see supplementary material. In each iteration, the Pairwise Similarity Learning module and Quantized Center Learning module are optimized in an alternating way to learn the continuous variable $\mathbf{U}$ and discrete variables $\{\mathbf{B}, \mathbf{C}\}$, respectively.

5 Experiments

Datasets. We conduct experiments on three datasets: CIFAR-10, NUS-WIDE and ImageNet100. CIFAR-10 consists of 60k color images with the resolution of 32×32 categorized into 10 classes. Each image has a single label. NUS-WIDE is a multi-label dataset, which contains 269,648 color images collected from Flickr. There are 81 classes, where each image is annotated with one or multiple class labels. Following [17, 19, 24], we use a subset of 195,834 images associated with 21 most frequent classes (concepts) for evaluation, among which 105,972 images has more than two labels and 89,862 images have a single label. Each class contains at least 5,000 samples. ImageNet100 consists of 130K single labelled images from 100 categories, which is a subset of the large benchmark ImageNet [6].

Experimental settings. Following [19, 20], 100 random images per class in CIFAR-10 form the test query set and 500 images per class are the training set. For NUS-WIDE, we randomly select 100 images per class as test queries and 500 images per class as the training set. The pairwise ground truth for two images sharing at least one common label is similar and otherwise dissimilar. Following [3], we sample 100 images per class for ImageNet100 to construct a training set, and all the images in the validation set are used as the test set.

Evaluation metrics. We evaluate retrieval performance using: mean Average Precision (MAP), precision of the top N returned examples (P@N), Precision-Recall curves (PR) and Recall curves (R@N). All compared methods use identical training and test sets for fair comparison. For NUS-WIDE, we adopt MAP@5000 and MAP@50000 for the small-data setting and large-data setting, respectively. We show the results of MAP@1000 for ImageNet100.

Network and parameter settings. To have a fair comparison with previous methods [19, 20, 32], we fine-tune the VGG-F[19, 20] architecture for the experiments on CIFAR-10 and NUS-WIDE while the AlexNet architecture [14] is fine-tuned for the experments on ImageNet100. Both deep network architectures are pre-trained on ImageNet. The hyper-parameters $\{\varphi, \mu, \eta, \nu, \}$ are tuned by cross-validation on a validation set and the margin m is chosen from $\{0.5, 1, 1.5, 2\}$. Stochastic Gradient Descent (SGD) is used for optimization.

5.1 Exp 1: Effect of Quantized Center Learning

To investigate the effect of L_{Intra} (minimizing intra-class distances) and L_{Inter} (maximizing inter-class distances) in the Quantized Center Learning module, we conduct an ablation study in the small-data setting which starts with the Pairwise Similarity Learning module L_{pair} in Eq. (3) in the model and then augment the model incrementally with L_{intra} in Eq. (4) and L_{Inter} in Eq. (7). In Table 1 we show the experimental results. We observe that both L_{Intra} and L_{Inter} contribute substantially to the performance of the whole model.

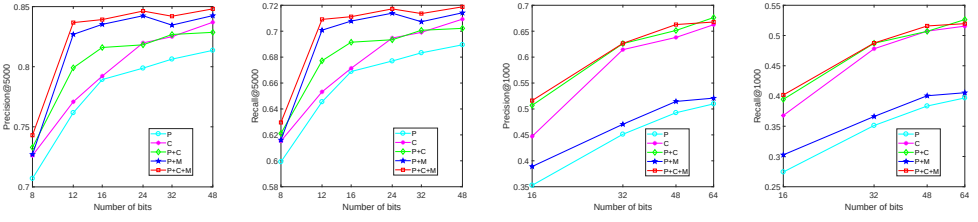
5.2 Exp 2: Functionality of different modules

We evaluate the effect of combining modules on both CIFAR-10 and ImageNet100 datasets using precision and recall curves for top 5,000 returned images for different number of bits. In Fig. 5 we compare on CIFAR-10 and ImageNet100. We observe that each module adds value. The only exception is Fisher-only, which outperforms the combined Pairwise+Fisher model for a code size of 48. Second, the combined models can get relatively well for fewer bits, while the single models need more bits to achieve the same performance.

The results on ImageNet100 shown in Fig. 5 indicate that the Quantized Center Learning module improves the performance substantially. One potential explanation is that the

Baseline	Components L_{Intra}	Components L_{Inter}	CIFAR-10 12 Bits	CIFAR-10 24 Bits	ImageNet100 16 Bits	ImageNet100 48 Bits
L_{pair}	×	×	0.730	0.787	0.431	0.572
	✓	×	0.746	0.802	0.543	0.696
	✓	✓	0.772	0.809	0.576	0.726

Table 1: Comparative results for our model with different components of the Quantized Center Learning module on CIFAR-10 and ImageNet100 . We start with the Pairwise Similarity Learning (L_{pair}) and augment incrementally with two components: L_{Intra} in Eq. (4) and L_{Inter} in Eq. (7). For 24-bits in CIFAR-10 the performance seems already saturated; for all other settings, each added component brings an advantage.



CIFAR-10 ImageNet100

Figure 5: Evaluating different modules on two datasets. Herein **P** refers to the Pairwise Similarity Learning module without margin while **C** refers to the Quantized Center Learning module. $\mathbf{P}+\mathbf{M}$ denotes the Pairwise Similarity Learning module with tuned margin.

Pairwise Similarity Learning module ($\tilde{L}_{\text{pair}}$) is sensitive to the balance between the positive and negative training sample pairs, which is hard to achieve in the data with large number of classes. In contrast, the Quantized Center Learning module does not suffer from this limitation. The sensitivity of the margin m is in the supplemental.

5.3 Exp 3: Comparison with others

In Table 2 we show results on both CIFAR-10 and NUS-WIDE datasets in the small-data setting. In particular for a few number of bits, our model compares well to others. It is worth noting that the performance comparison among VGG-F and AlexNet networks is considered to be fair [31], since both architectures have the same network composition.

Method	CIFAR-10 12 bits	CIFAR-10 24 bits	CIFAR-10 32 bits	CIFAR-10 48 bits	Method	NUS-WIDE 12 bits	NUS-WIDE 24 bits	NUS-WIDE 32 bits	NUS-WIDE 48 bits
Ours	**0.803**	**0.825**	**0.831**	**0.844**	Ours	**0.795**	**0.823**	**0.833**	**0.842**
DSDH [19]	0.740	0.786	0.801	0.820	DSDH [19]	0.776	0.808	0.820	0.829
Greedy Hash [31]	0.774	0.795	0.810	0.822	Greedy Hash [31]	–	–	–	–
DPSH [20]	0.713	0.727	0.744	0.757	DPSH [20]	0.752	0.790	0.794	0.812
DQN [3]	0.554	0.558	0.564	0.580	DQN [3]	0.768	0.776	0.783	0.792
DTSH [33]	0.710	0.750	0.765	0.774	DTSH [33]	0.773	0.808	0.812	0.824
NINH [18]	0.552	0.566	0.558	0.581	NINH [18]	0.674	0.697	0.713	0.715
CNNH [32]	0.439	0.511	0.509	0.522	CNNH [32]	0.611	0.618	0.625	0.608

Table 2: MAP for various methods for the small-data setting for CIFAR-10 and NUS-WIDE. The best performance is boldfaced. For NUS-WIDE, the top 5,000 is used for the MAP.

The state-of-the-art DSDH [19] model also uses pairwise labels and classification labels. The major difference between is in using the classification label: DSDH [19] learns hash codes by maximizing the classification performance while our model learns centers to model between-class and between-sample distances. While DSDH performs excellent, our model outperforms DSDH in all experiments.

Another interesting observation is that SDH [30], which is based on sole classification label information, performs competitively on NUS-WIDE but not as good on CIFAR-10. In contrast, our model and DSDH [19] that leverage two types of information, perform much more robust. It reveals the necessity of incorporating the pairwise label information.

We also conduct experiments to compare our method to other baseline models on ImageNet100 and the results are presented in Table 3. It is observed that our model achieves the best performance on all bits except for the 16 bits.

	ImageNet100 (mAP@1K)			
Method	16 Bits	32 Bits	48 Bits	64 Bits
CNNH [34]	0.281	0.450	0.525	0.554
NINH [18]	0.290	0.461	0.530	0.565
DHN [39]	0.311	0.472	0.542	0.573
HashNet [3]	0.506	0.630	0.663	0.683
Greedy Hash [31]	**0.625**	0.662	0.682	0.688
Ours	0.590	**0.697**	**0.726**	**0.747**

Table 3: MAP@1K results on ImageNet100 using AlexNet.

6 Conclusion

We present a supervised deep binary hashing method focusing on binary separability through a pair-wise margin and inspired by Fisher's linear discriminant which minimizes within-class distances while maximizing between-class distances. For medium-sized datasets with much training data –where larger hash codes can be used– our method performs on par or only slightly better than other methods. Our method is most suitable for extremely large datasets with few training data where only tiny bit codes can be used; there our method compares most favorably to others.

References

[1] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, 2016.

[2] Yue Cao, Mingsheng Long, Liu Bin, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *CVPR*, 2018.

[3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. 2017.

[4] Shih Fu Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[5] Mayur Datar and Piotr Indyk. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 253–262. ACM Press, 2004.

[6] Jia Deng, Wei Dong, R Socher, and Li Jia Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[7] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of International Conference on Very Large Databases*, pages 518–529, 2000.

[8] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[9] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis Machine Intelligence*, PP(99):1–1, 2018.

[10] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[11] Qing Yuan Jiang and Wu Jun Li. Scalable graph hashing with feature transformation. In *International Conference on Artificial Intelligence*, 2015.

[12] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. In *CVPR*, 2017.

[13] Weihao Kong and Wu Jun Li. Isotropic hashing. In *NIPS*, 2012.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[15] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[16] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 31(12):2143, 2009.

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

[18] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015.

[19] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *NIPS*. 2017.

[20] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 2016.

[21] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van Den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[22] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. *CVPR*, 2016.

[23] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Learning multifunctional binary codes for both category and attribute oriented retrieval tasks. In *CVPR*, 2017.

[24] Wei Liu, Jun Wang, and Shih fu Chang. Hashing with graphs. In *ICML*, 2011.

[25] Wei Liu, Sanjiv Kumar, Sanjiv Kumar, and Shih Fu Chang. Discrete graph hashing. In *NIPS*, 2014.

[26] Yadong Mu and Shuicheng Yan. Non-metric locality-sensitive hashing. In *AAAI*, 2010.

[27] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[28] M Raginsky. Locality-sensitive binary codes from shift-invariant kernels. 2009.

[29] Ramin Raziperchikolaei and Miguel Á Carreira-Perpiñán. Optimizing affinity-based binary hashing using auxiliary coordinates. In *NIPS*, 2016.

[30] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[31] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Advances in Neural Information Processing Systems*, pages 798–807, 2018.

[32] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. *Asian Conference on Computer Vision*, 2016.

[33] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, 2008.

[34] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014.

[35] Ting Yao, Fuchen Long, Tao Mei, and Yong Rui. Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*, 2016.

[36] Peichao Zhang, Wei Zhang, Wu Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *SIGIR*, 2014.

[37] Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient training of very deep neural networks for supervised hashing. In *CVPR*, 2016.

[38] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[39] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, 2016.