Pattern Recognition 86 (2019) 134–142

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Bayesian denoising hashing for robust image retrieval

Dong Wang, Ge Song, Xiaoyang Tan*

Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

ARTICLE INFO

Article history:
Received 8 December 2017
Revised 2 August 2018
Accepted 9 September 2018
Available online 10 September 2018

Keywords:
Image retrieval
Denoising hashing
Probabilistic model
Variational Bayes

ABSTRACT

Learning to hash is one of the most popular techniques in image retrieval, but few work investigates its robustness to noise corrupted images in which the unknown pattern of noise would heavily deteriorate the performance. To deal with this issue, we present in this paper a Bayesian denoising hashing algorithm whose output can be regarded a denoised version of the input hash code. We show that our method essentially seeks to reconstruct a new but more robust hash code by preserving the original input information while imposing extra constraints so as to correct the corrupted bits. We optimized this model in variational Bayes framework which has a closed-form update in each iteration that is more efficient than numerical optimization. Furthermore, our method can be added at the top of any original hashing layer, serving as a post-processing denoising layer with no change to previous training procedure. Experiments on three popular datasets demonstrate that the proposed method yields robust and meaningful hash code, which significantly improves the performance of state-of-the-art hash learning methods on challenging tasks such as large-scale natural image retrieval and retrieval with corrupted images.

1. Introduction

With the popularization of high-pixel camera phones and the explosive growth of the Internet, massive images have flooded our daily lives. Image retrieval, i.e., finding images containing the same object or scene as in a query image, has attracted much attention from both academia and industry in recent years. And for this task, the semantic gap between low-level content and higher-level concepts remains the major challenge for all current approaches that rely on visual similarity for judging semantic similarity [1]. Traditional methods extracting visual content from images are mainly based on statistical information at the pixel level, e.g., [2–7]. Although these methods have shown the robustness against certain types of noise, low-level representation can hardly capture semantic information, which limits its effectiveness in real-world applications.

Recently, many studies have shown that Deep Convolutional Neural Networks (CNNs) could learn representations with high-level semantic concepts and reported that it achieved the state-of-the-art performance in many computer vision tasks such as image recognition [8–10] object detection [11] or semantic segmentation [12]. Notably, in image retrieval field, many works adopted solutions based on features extracted from a CNN pre-trained for the task of image classification [8,13,14]. Most of these [15–17] paid more attention on features from the deep convolutional layers of CNNs, and demonstrated that they contain particular semantic meaning of local image regions and hence lead to better performance. However, the CNN features are faced with the problem of high computational cost in similarity calculation due to the high-dimensional representation, hence being not appropriate for the task of efficient nearest neighbor search in image retrieval.

A practical solution to this issue is the hash method, which aims to learn a nonlinear function that transforms the real-valued features to much compact binary codes such that similar data items are mapped into similar codes [18–30]. In fact, it has become one of the most popular storage and retrieval approaches in handling large-scale vision problems [31–34]. The learned hash codes can be either used to index data items or to approximate the distance in nearest neighbor search. The goal of hash coding can be regarded as a type of representation learning, and it can be implemented in either supervised or unsupervised way. Supervised methods [35–39] use label information to regularize the behavior of learned hash codes such that they help to pull together samples from the same class while pushing away those from different classes. Besides supervised hashing, unsupervised [40–43] and semi-supervised [44,45] hash methods also enjoy their popularity in certain applications.

Despite the success, the robustness property of hash codes has seldom been studied. The reasons of the lack of robustness of hash codes could be mixed with various factors such as the limitation of

* Corresponding author.
E-mail addresses: dongwang@nuaa.edu.cn (D. Wang), sunge@nuaa.edu.cn (G. Song), x.tan@nuaa.edu.cn (X. Tan).

https://doi.org/10.1016/j.patcog.2018.09.006
0031-3203/© 2018 Elsevier Ltd. All rights reserved.

the underlying optimization methods, or the inevitable information loss due to feature binarization. Due to these, the hash codes are usually vulnerable to the noise in input images, hence deteriorating the retrieval performance. Previous work deals with this issue either by explicitly identifying the suspicious occluded regions in the input image through occlusion detection [46–48], or by adopting specially designed similarity function [49–51]. However, these methods have not been used in the context of hash learning. Some researchers also try to use deep neural network [52,53] to learn robust representation, but when the noise level is relatively large, the CNN representation would be corrupted by itself, not mention the subsequent hash code.

There are many traditional methods on learning a compact and robust representation, However, they almost focus on continuous data that can not properly deal with binary data. For example, using robust loss function [54–56] or introducing regularization [57–59], or employing robust optimization [60,61]. However, these methods are not fit for binary data. Neural network based models, e.g., denoising autoencoder (DAE) [62] and denoising Restrict Boltzmann Machine (RBM) [63] which are variants of AE and RBM respectively, can be used to learn denoising feature representation on both continuous and binary data which is more useful in real-world applications. However, the unsupervised learning algorithms can not learn the optimal representation on labeled data. Recently, deep learning based denoising methods [64,65] have been proposed, however they also work on natural images not binary input and have enormous computation cost.

In this paper, we propose a Bayesian denoising hashing that serving as a post-processing denoising layer to improve the robustness of existing hash codes. We show that our method essentially seeks to reconstruct a new but more robust hash code by preserving the original input information while imposing extra constraints so as to correct the corrupted bits. We optimized this model in variational Bayes framework, and in each training iteration it has a closed-form update which is more efficient than numerical optimization. It is worth noting that our method can be added at the top of any original hashing layer with no change to previous training procedure. Experiments on several benchmark datasets demonstrate that the proposed method yields robust and meaningful hash code, which significantly improves the performance of state-of-the-art hash learning methods on challenging tasks such as large-scale natural image retrieval and retrieval with corrupted images.

The remaining parts of this paper are organized as follows: We detail our method in Section 2. In Section 3, we investigate the performance of our method empirically over several popular datasets. We conclude this paper in Section 4.

2. The proposed method

2.1. Bayesian denoising hashing

In this section, we will detail the proposed method. Fig. 1 illustrates the whole pipeline. There are three steps: The first is to extract discriminant feature representation via well pretrained CNN model (e.g., Deep Residual Network [66]). Next, a state-of-the-art hash model (e.g., Fast Hash (FastH) [37] is trained on these data. Finally, we use our Bayesian denoising hashing method to correct the possible error bits resulted from the second step.

Formally, assume that we have a binary dataset $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ where $x_i \in \{0, 1\}^D$ is a D-dimension hash vector and $y_i \in \{1, 2, \dots, K\}$ is the class assignment for x_i (K is the number of classes). Furthermore, we assume that these hash codes x_i are corrupted due to unknown disturbance, but their label information is clean, and our goal is to reconstruct a denoised version of them. This target is similar to the Denoising Autoencoder [62] but we do not know the clean version of the input.

We introduce a latent variable z_i for each observation (x_i, y_i), with its dimension identical to that of x_i. We can think of z_i as the continuous version of the final denoised hash code which can be obtained simply by thresholding the corresponding z_i. On one hand, we use each dimension z_i^d of z_i to control the corresponding hash bit x_i^d, and on the other hand, we expect those z_i from the same class to be close to each other while those from different classes to be kept far away. Consequently, the final value of each z_i should be a tradeoff of these two constraints. To do this, we model the joint distribution of the interested random variable (or random vector) X, Y and Z as follows,

$$p(X, Y, Z) \quad = \int_W p(X|Z)p(Y|Z, W)p(Z)p(W)dW \tag{1}$$

where W is the global parameter that increases the flexibility of the model, and we also assume that X and Y are conditionally independent given latent code Z. Also note that although in principle we can parameterize the generative model $p(X|Z)$, in this work for simplicity we exploit the independence assumption made above to get,

$$p(X|Z) \quad = \prod_i p(x_i|z_i) = \prod_i \prod_d p(x_i^d|z_i^d) \tag{2}$$

For convenience, if the binary variable x_i^d is 0, we convert it to -1. Hence $x_i^d \in \{-1, 1\}$, which allows us to model $p(x_i^d|z_i^d)$ directly using logistic regression:

$$p(x_i^d|z_i^d) = \theta(x_i^d z_i^d) \tag{3}$$

where $\theta(t)$ is the sigmoid function:

$$\theta(t) = \frac{1}{1+\exp(-t)} \tag{4}$$

In this way, z_i^d controls the corresponding hash bit x_i^d, i.e., a positive z_i^d implies $x_i^d = 1$ with high probability, and $x_i^d = -1$ otherwise.

To identify behaviorally relevant variables, we have to assume a supervised learning setting. Instead of modeling the dependent variable Y directly, we factorize the recognition model $p(Y|Z, W)$ into a set of pairwise constraints:

$$p(Y|Z, W) \quad = \prod_{ij} p(y_{ij}|z_i, z_j, W) \tag{5}$$

where $y_{ij} = 1$ if $y_i = y_j$, and $y_{ij} = -1$ otherwise. $W \in R^{D\times D}$ is the global parameter (D is the dimension of the latent code Z). We parameterize the pairwise model $p(y_{ij}|z_i, z_j, W)$ such that it behaves according to the constraints mentioned at the beginning of this section. Specifically, it is defined as follows:

$$p(y_{ij}|z_i, z_j, W) \quad = \theta(y_{ij} z_i^T W z_j) \tag{6}$$

Hence, our method is essentially a trade-off between preserving the original input information and imposing extra supervised constraints to correct the possible error bits. Also note that we only perform this method on the gallery set, and we do not change the hash bits of query data due to that their class information is usually unknown. We show in the experiments that refining the hash code is very helpful for better image retrieval when the gallery set is corrupted by big noise.

Next we will detail how to learn the global parameter W and to infer the latent variable Z given X.

2.2. Variational inference

First we choose a prior distribution for each z_i, e.g. Gaussian distribution, for its simpleness and smoothing property.

$$p(z_i) = \quad \mathcal{N}(z_i|\vec{0}, \sigma_z^2 I) \tag{7}$$

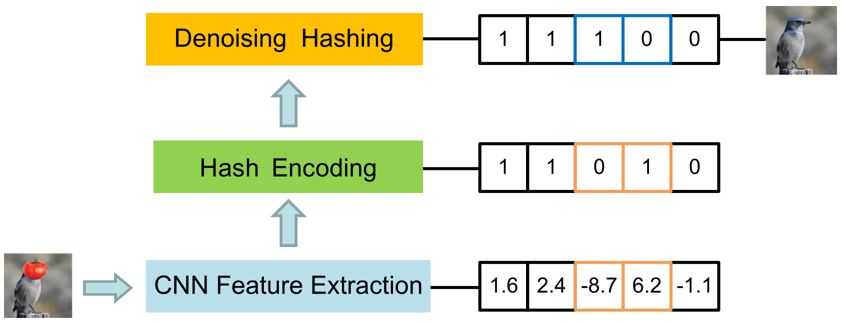


Fig. 1. Hash codes learned with CNN features may be distracted by some types of noise such as partial occlusion (e.g., here the head of the bird is occluded.). Our Bayesian denoising hashing can be stacked at the top of the original hashing layer so as to improve its robustness against noise. Right (conceptual illustration): corrupted hash bits (shown in the yellow boxes of the bottom two rows) due to occlusion are corrected by our method (shown in the blue boxes at the top row). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Then with the Bayes rule, we obtain the posterior distribution of z_i,

$$p(z_i| \quad x_i, Y, Z_{N_i}, W) = \frac{p(x_i|z_i)\prod_{j\in N_i} p(y_{ij}|z_i,z_j,W)p(z_i)}{\int_{z_i} p(x_i|z_i)\prod_{j\in N_i} p(y_{ij}|z_i,z_j,W)p(z_i)dz_i} \tag{8}$$

where $N_i = \{j|y_j = y_i\}$ is the set of indices of all points have label y_i, and $Z_{N_i} = \{z_j|j \in N_i\}$ includes all the pairwise constraints of z_i constructed with N_i. For the sake of efficiency, we can randomly sample a small subset in each N_i (e.g. let $|N_i| = 5$ for each i) instead of using all the constraints.

The posterior distribution of z_i is intractable due to the fact that 1) each z_i is coupled with several z_j, $j \in N_i$, and 2) the likelihood terms $p(x_i|z_i)$ and $p(y_{ij}|z_i, z_j, W)$ are not conjugate to the prior $p(z_i)$. Hence we approximate the posterior using a distribution $q(z_i)$ within the variational inference (VI) framework [67]. Using the same method, we also estimate the model parameter W, and the main results are summarized in the following theorem:

Theorem 1. *The approximated posterior distribution $q(z_i)$ to Eq. (8) by the variational inference is given by,*

$$\begin{aligned} q(z_i) &= \mathcal{N}(z_i|\bar{z}_i, V_{z_i}) \\ V_{z_i} &= [\sigma_z^{-2}I + 2\textstyle\sum_{j\in N_i}\lambda(\xi_{z_{ij}})E(z_j^T W^* W^* z_j) + 2\Lambda_i]^{-1} \\ \bar{z}_i &= \tfrac{1}{2}V_{z_i}[x_i + \textstyle\sum_{j\in N_i}(2y_{ij}-1)E(W^* z_j)] \end{aligned} \tag{9}$$

and the optimal global matrix W^* is recovered from its vectorized version w^*, given by,

$$w^* = [V_{w_0}^{-1} + 2\textstyle\sum_{ij}\lambda(\xi_{z_{ij}})E(z_{ij}z_{ij}^T)]^{-1}[V_{w_0}^{-1}\mu_{w_0} + \textstyle\sum_{ij}(y_{ij} - \tfrac{1}{2})E(z_{ij})] \tag{10}$$

where $E(\cdot)$ is the expectation operator, $\lambda(x) = -\frac{1}{2x}[\theta(x) - \frac{1}{2}]$, $\Lambda_i =$ diag$(\Lambda_i^1, \Lambda_i^2, \ldots, \Lambda_i^D)$ is a diagonal matrix with $\Lambda_i^d = \lambda(\xi_{z_i^d})(x_i^d)^2$, $\mathcal{N}(w|\mu_{w_0}, V_{w_0})$ is the prior distribution for the model parameter W ($w = \mathrm{vec}(W)$), $z_{ij} = \mathrm{vec}(z_i z_j^T)$.[1] $\xi_{z_i^d}$ and $\xi_{z_{ij}}$ are the variational parameters.

Proof. According to the variational inference, we seek to maximize the following variational lower bound,

$$\max \ L(q) = \int q(Z)\ln\frac{p(X,Y,Z,w)}{q(Z)}dZ \tag{12}$$

This leads to the following solution,

$$\ln q(z_i) = E_{-z_i}[\ln p(X,Y,Z,w)] \tag{13}$$

$$= E_{-z_i}[\ln p(X|Z)P(Y|Z,w)P(Z)p(w)] \tag{14}$$

Now we adopt a factorized distribution $q(Z) = \prod_{i=1}^N q(z_i)$ to approximate true posterior distribution of Z, and plug Eqs. (3) and (6) into Eq. (14) and ignore those terms irrelevant to z_i, we obtain,

$$\ln q(z) = E_{-z_i}[\textstyle\sum_d \ln\theta(x_i^d z_i^d) + \sum_{j\in N_i}\ln\theta(y_{ij}z_i^T W z_j) + \ln p(z_i)] \tag{15}$$

To make the likelihood conjugate to the prior, we follow Jaakkola and Jordan [68] to approximate the logistic sigmoid function appeared in the right hand of Eq. (15) with its quadratic lower bound. This leads to the following approximation of the log posterior of interest,

$$\begin{aligned} \ln q(z_i) = \quad & -\textstyle\sum_{d=1}^D\{\lambda(\xi_{z_i^d})[(x_i^d z_i^d)^2 - \xi_{z_i^d}^2] + \frac{1}{2}(x_i^d z_i^d + \xi_{z_i^d})\} \\ & -\textstyle\sum_{j\in N_i}E_{-z_i}\{\lambda(\xi_{z_{ij}})[(y_{ij}z_i^T W z_j)^2 - \xi_{z_{ij}}^2] \\ & +(\tfrac{1}{2} - y_{ij})z_i^T W z_j + \tfrac{1}{2}\xi_{z_{ij}}\} - \tfrac{1}{2}\sigma_z^{-2}z_i^T z_i + \text{const} \end{aligned} \tag{16}$$

Then with some mathematical manipulations, we obtain the posterior distribution $q(z_i)$ as shown in Eq. (9). □

Next we turn to estimate parameter W. We first vectorize the matrix W and $z_i z_j^T$ w.r.t. w and z_{ij}, as shown in Eq. (11). With this, we reformulate the function of W (cf., Eq. (6) as an inner product:

$$z_i^T W z_j = w^T z_{ij} \tag{17}$$

After obtaining the MAP (maximum a posterior) solution w_*, we reshape w_* back to W_*. From Eq. (12), the objective of w is,

$$\max \ L_w = \int q(Z)\ln p(X,Y,Z,w)dZ \tag{18}$$

By introducing the Gaussian prior $\mathcal{N}(w|\mu_{w_0}, V_{w_0})$ of w and using the Jaakkola bound [68] again, we get,

$$\begin{aligned} L_w \geq \quad & -\textstyle\sum_{ij}E_Z\{\lambda(\xi_{z_{ij}})[(y_{ij}w^T z_{ij})^2 - \xi_{z_{ij}}^2] \\ & +(\tfrac{1}{2} - y_{ij})w^T z_{ij} + \tfrac{1}{2}\xi_{z_{ij}}\} \\ & -\tfrac{1}{2}(w - \mu_{w_0})^T V_{w_0}^{-1}(w - \mu_{w_0}) + \text{const} \end{aligned} \tag{19}$$

Taking the partial derivative $\partial L_w/\partial w$ and set it to zero, we can get the closed-form solution — w^* (or W^*) (as in Eq. (10)). Furthermore, according to Jaakkola and Jordan [68], the variational parameters $\xi_{z_i^d}$, $\xi_{z_{ij}}$ can be estimated as follows:

$$\begin{aligned} \xi_{z_i^d} &= \sqrt{E[(z_i^d)^2]} \\ \xi_{z_{ij}} &= \sqrt{E[(z_i^T W^* z_j)^2]} \end{aligned} \tag{20}$$

This completes the proof of Theorem 1.

[1]

$$\begin{aligned} z_{ij} = \quad & [z_i^1 z_j^1, z_i^1 z_j^2, \ldots, z_i^1 z_j^D, z_i^2 z_j^1, z_i^2 z_j^2, \ldots, z_i^2 z_j^D, \ldots, z_i^D z_j^1, z_i^D z_j^2, \ldots, z_i^D z_j^D]^T \\ w = \quad & [W_{11}, W_{12}, \ldots, W_{1D}, W_{21}, W_{22}, \ldots, W_{2D}, \ldots, W_{D1}, W_{D2}, \ldots, W_{DD}]^T \end{aligned} \tag{11}$$

During training, we iteratively update latent variable z_i, global parameter w (or W) and variational parameter ξ_{z_i}, $\xi_{z_{ij}}$. In each iteration, the computational cost of learning Z is $o(ND)$ and of learning W is $o(ND^3)$. This may seem big, but note that the input of our method is binary hash code from a state-of-the-art hashing algorithm which is usually very compact, i.e., no more than 100 bits. And usually this closed-form updating just needs very a few training iterations to converge to a local optima.

During inference, we use $q(z_i)$ to deduce the groundtruth value of x_i. Let us define $\tilde{x}_i^d$ as the corrected bit. Then the relationship between $\tilde{x}_i^d$ and $\bar{z}_i^d$ (Eq. (9)) is:

$$\tilde{x}_i^d = \begin{cases} -x_i^d & \text{if } x_i^d \neq \text{sign}(\bar{z}_i^d) \text{ and } |\bar{z}_i^d| > \text{threshold} \\ x_i^d & \text{otherwise} \end{cases} \tag{21}$$

Note that this is a conservative strategy that the introduced threshold is used to correct those bits whose margin (i.e., $x_i^d \bar{z}_i^d$) is smaller than some predefined value [2]. We summarize the proposed method in Algorithm 1 .

Algorithm 1 Bayesian denoising hashing.

```
Input:
    Input: Noisy hash dataset $\{(x_i, y_i)|i = 1, 2, ..., N\}$, prior distribution $p(z_i), i = 1, 2, ..., N$ and $p(w) = \mathcal{N}(w|\mu_{w_0}, V_{w_0})$.
Output:
    Denoised dataset $\{(\tilde{x}_i, y_i)|i = 1, 2, ..., N\}$
    —— Training Stage
1:  Initialize $q(z_i)$, $(i = 1, 2, ..., N)$ with prior $p(z_i)$, set $w$ to $\mu_{w_0}$; then compute $\xi_{z_i^d}$, $\xi_{z_{ij}}$ with eq. (20) respectively.
2:  Repeat
3:      update $q(z_i)$, $(i = 1, 2, ..., N)$ with eq. (9).
4:      update $w$ with eq. (10).
5:      update $\xi_{z_i^d}$, $\xi_{z_{ij}}$ with eq. (20).
6:  Until converged.
7:  Detect and correct the error bits in $x_i$, $(i = 1, 2, ..., N)$ with eq. (21).
8:  Return denoised dataset $\{(\tilde{x}_i, y_i)|i = 1, 2, ..., N\}$.
```

3. Experiments

3.1. Setting

To evaluate the performance of the proposed method, we conduct a series of experiments on three well-known datasets: CIFAR-10 [69], MNIST [70], and ImageNet [71].

On each dataset we use random noise to synthesize corrupted images, particularly, we blurred a randomly selected square region with its mean value for each training image of a given dataset. How many images are undertaken such corruption depends on the specific evaluation protocol of a given dataset - we only do this on the training partition (including the gallery set or the set to be retrieved) of the dataset while the query images are kept clean.

All the input images undergo whitening and contrast normalization as preprocessing steps [69]. We use the mean average precision (MAP) as evaluation metric to report the retrieval performance both before and after adding our Bayesian denoising hashing layer.

3.2. Experiments on the MNIST dataset

We first investigate the behavior of our methods on MNIST dataset. MNIST [70] is a large scale digital recognition dataset containing 70,000 28×28 gray images (10 categories, training/test : 60,000/10,000). On this dataset, we add occlusion noise to the gallery set by varying the occlusion size from the 4×4 to 20×20 in pixel. Fig. 2 illustrates some of the examples. We can see that when the occlusion size is small (4×4), it has little effect on recognition while as the size increases to 20×20 it is even impossible for human to recognize. On this dataset, we apply the proposed Bayesian denoising hashing method on three baseline hashing models: Kernel-Based Supervised Hash (KSH) [35], Fast Hash (FastH) [37] and Supervised Discrete Hash (SDH) [38]. For each of them, we test three hash code lengths — 12-bits 24-bits and 48-bits. We use the C-SVDDNet model [72] for feature extraction.

Fig. 3(a)–(c) give the results, where the performance of various hash methods before and after adding our Bayesian denoising layer ('+Bayes') is reported. We can see that under 48 bits code length and no occlusion, the current state-of-the-art hash method like KSH, SDH, FastH achieves excellent performance of 99.3% 99.0% 99.6% MAP respectively. However, as the degree of random structural distortion increases, the performance of all methods decreases quickly. Particularly the performance of the FastH method [37] decreases about 3.0%, that is, from 99.6% to 96.6%, is the most robust one among them, while the performance of all the other two methods decreases more than 15.0%. The situation becomes even worse if a shorter code is adopted. For example, under 12 bits, even the FastH decreases nearly 9.0% MAP from 99.6% to 90.7%. From these observations we conclude that random partial occlusion on the images does impose a challenge onto the current learning to hash methods.

Fortunately, Fig. 3(a)–(c) also reveal that when equipped with our Bayesian denoising layer, the performances of all the baseline methods are consistently improved significantly regardless of the code length. For example, our method remarkably improves the performance of the KSH [35] and SDH [38] by more than 15.0% at the occlusion size of 20×20 in pixels. Even for the most stable FastH method, adding our Bayesian denoising layer could lead to about 4.0% performance improvement at the code length of 12 bits.

By treating the hash code obtained with clean data as ground truth, we can compare it with the hash code generated at various size of partial occlusion to gain more understanding of the behavior of these methods. Fig. 4 shows the error statistics about how many hash bits are erroneously flipped due to partial occlusion. One can see that as the occlusion size increases, more bits are flipped by the FastH method, but with our Bayesian postprocessing layer many erroneous bits are corrected. This is consistent with and partly explains the results of Fig. 3(c).

To show the efficiency of the our variational Bayesian training, we also conduct experiments on MNIST dataset by varying the number of training iterations of our method. Fig. 5 shows the results that even with big occlusion, our method only needs a few iterations (3 to 5) to converge to a local minima for all the three hashing methods.

3.3. Experiments on the CIFAR-10 dataset

We next evaluate our method on the famous CIFAR-10 dataset which contains 60,000 32×32 color natural images (10 categories, training/test : 50,000/10,000). On this dataset, besides random occlusion we have considered two other types of occlusions, i.e., occlusion on the object and occlusion on the background. The simulation of the corruption is made by blurring either the center region of a given image (for occlusion on the object) or the corners of it (for occlusion on the background), see Fig. 6 for illustration. Over these images, we apply the proposed Bayesian denoising hashing post-processing on two types of hashing: one is the end-to-end deep hashing, e.g., part based deep hashing (PDH)

[2] By default, in our experiments we set the threshold to be 1 unless otherwise noted.

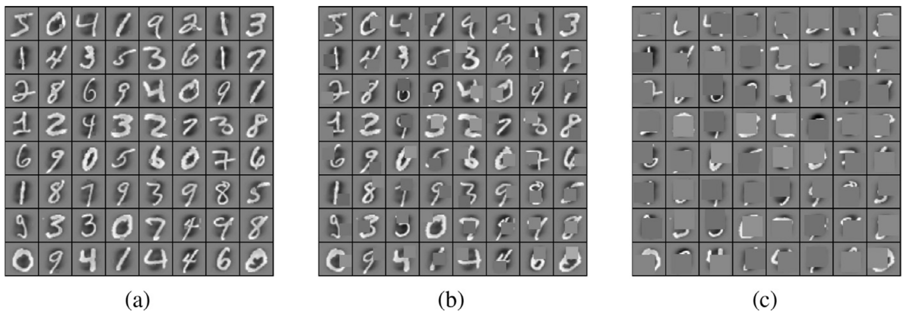


Fig. 2. Illustration of the images of MNIST dataset with varying degrees of partial occlusion. The size of occlusion is (a) 4×4, (b) 12×12, (c) 20×20.

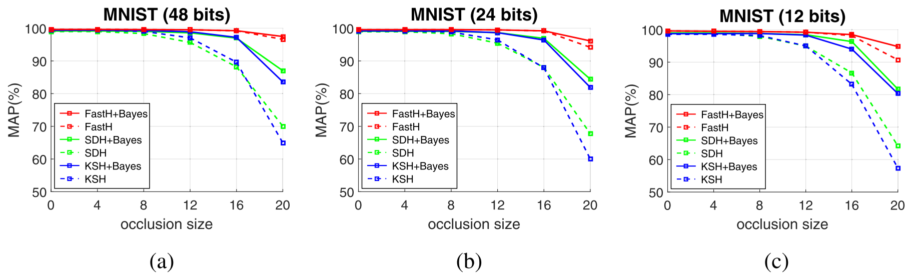


Fig. 3. Comparative Performances of various deep hash methods for image retrieval on the MNIST datasets.

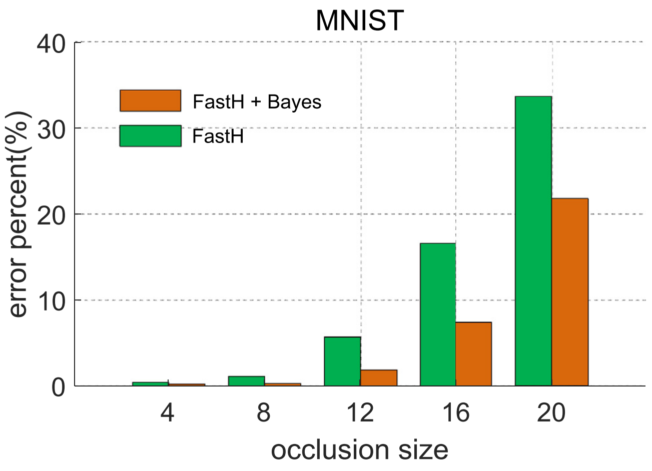


Fig. 4. Comparison of the proportion of erroneously flipped hash bits (%) with the FastH method and the proposed method under various size of partial occlusion on the MNIST dataset with code length 12 bits.

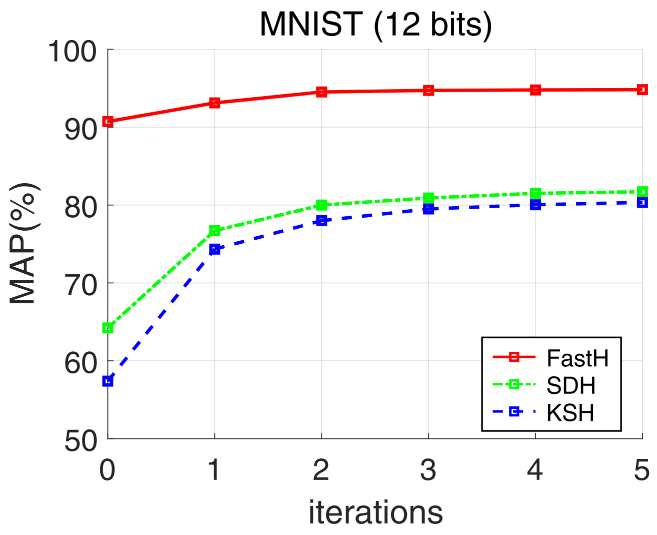


Fig. 5. Performance of the Bayesian denoising hashing by varying the number of training iterations (with 20×20 occlusion, code length = 12).

[33], semi-supervised deep hashing (SSDH) [26], Deep Weighted Hashing (DWH) [27], deep supervised hashing with pairwise labels (DPSH) [24]; the other is non-end-to-end hashing in which we first use a pretrained deep network (Wide ResNet [73]: 95.5% acc) to extract feature representations and then on it we train the shallow models such as FastH [37], SDH [38], KSH [35] and supervised hashing with latent factor models (LFH) [23]. The hash length for all the methods is set to be 24 bits. The other setting is the same as MNIST.

Fig. 7 gives the results. Both types of methods are sensitive to the occlusion noise. Particularly, the performance could be heavily deteriorated if the object to be retrieved is suffered from partial occlusion, e.g., the performance of LFH decreases from 94.25% to 66.29% with occlusion on the object, but with occlusion on background its performance remains to be high — 92.94%. Fortunately, in all these cases, our Bayesian denoising hashing can effectively improve the robustness against occlusion for both end-to-end hashing methods and non-end-to-end methods. For example, our method improves the performance of LFH from 66.29% to 77.92% in the object-corrupted case.

The results also show that sometimes the performance of non-end-to-end hashing methods may work better than end-to-end deep hashing methods. For example, the FastH model trained on a pretrained deep residual network achieves 93.43% MAP (with random occlusion noise) which outperforms all the other compared methods. This is possibly due to the reason that although end-to-end training helps to exploit the training data, it has to make a compromise between feature representation and prediction accuracy, which may hurt its generalization capability especially when the input data is noisy.

Fig. 8 gives the training time of our method. For those shallow hashing algorithms, adding our Bayesian post-processing will increase the training time by about 20–30% on average, e.g., for the KSH [35] method, the extra training time we pay for is about

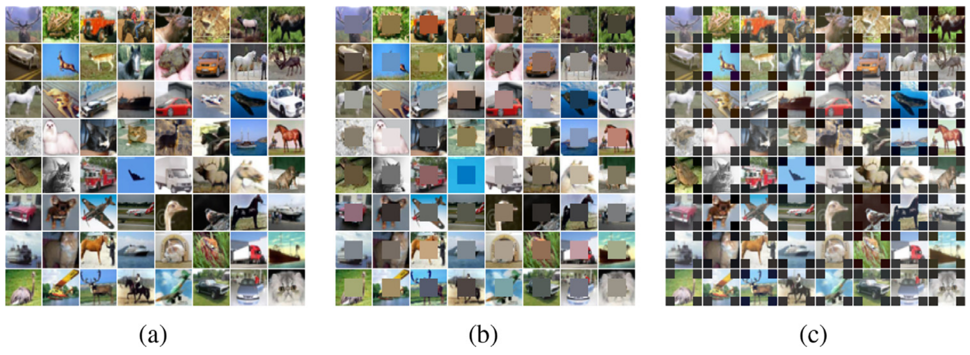


Fig. 6. Illustration of the images of CIFAR-10 dataset: (a) original images, (b) with occlusion on object, (c) with occlusion on background.

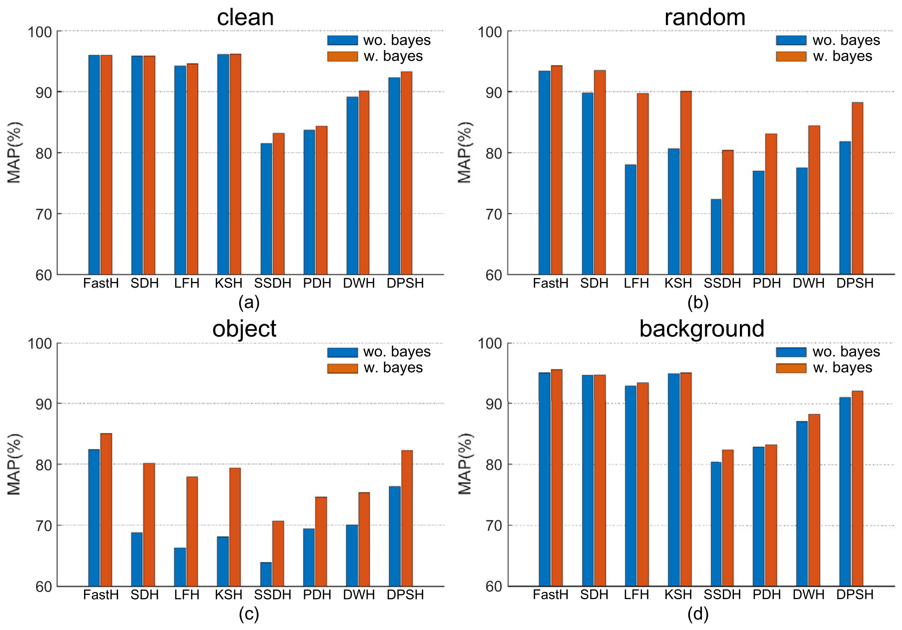


Fig. 7. Comparative performance (mAP %) on CIFAR-10 dataset (24-bits code). (a) clean images (b) with random occlusion (c) with occlusion on object (d) with occlusion on background.

3 min (i.e., the total training time increases from 14 to 17 min), but its robustness performance against random occlusion increases from 80.5% to 90.1%.

We also conduct an experiment to evaluate the effect of fintuning deep features by noise images. Particularly, we tested this method based on the LFH method [23]. Fig. 9 gives the results. It shows that finetuning helps to improve the robustness against occlusion, by increasing the performance of LFH from 78.04 to 80.21%. However, the effect of finetuning over LFH + Bayes is marginal, possibly due to the fact after our Bayes post-processing the resulted hashing codes have higher tolerance against noisy images.

Additionally, we conducted another experiment on CIFAR-10 dataset to investigate the effect of network structure. We test two networks, one is a pretrained Wide ResNet—WRN-16-8 [73], the other is the CNN-F network which we have finetuned on CIFAR dataset. Fig. 10 shows the performance of 4 shallow hashing methods (FastH [37], SDH [38], LFH [23], KSH [35]) with the WRN and CNN-F respectively under 16×16 occlusion noise. With our Bayesian denoising hashing, the performance of the 4 compared methods are significantly improved under both of the two network structure. Moreover, this result also shows that the performance of hashing does rely on the network structure. Using a deeper network can further improve the performance.

3.4. Experiments on the ImageNet dataset

Our final series of experiments are conducted on the large scale ImageNet database [71]. The database contains over 1.2 million color images of totally 1000 categories. We sample a subset of 100,000 images from ILSVRC2012 (1000 categories with 100 images per category) as the database to be retrieved and use the original validation partition as query set (50,000 images). As required by the ResNet-50, all the input images are resized to 224×224. Since many hash methods are not easily adaptive to large scale dataset due to its huge time complexity, on this dataset we only compare the performance of FastH, 'FastH + Bayes' and 'FastH + DAE'. The DAE (Denoising autoencoder [62]) method is a popular representation method that learns the manifold of dataset by trying to reconstruct the original data from its corrupted version. Similar to our Bayesian denoising layer, we use DAE to post-process the hash codes generated by the FastH, and compare its performance with ours.

Table 1 gives the results obtained with/without partial occlusion. As expected, one can see that for a large dataset like this a longer hash code is preferred. It is interesting to see that on this dataset, even under no any simulated random structural distortion, our Bayesian denoising layer significantly improve the performance of the original FastH method. For example, our method

Table 1
Comparative performance (mAP %) on ImageNet.

Code length	Clean				80 × 80 occlusion			
	24	48	1024	2048	24	48	1024	2048
FastH	4.69	14.02	57.7	60.8	2.46	9.49	53.0	56.7
FastH + DAE	3.68	12.54	53.5	56.1	2.01	8.37	48.7	51.2
FastH + Bayes	**12.44**	**27.98**	**66.0**	**67.8**	**7.45**	**25.47**	**63.6**	**65.6**

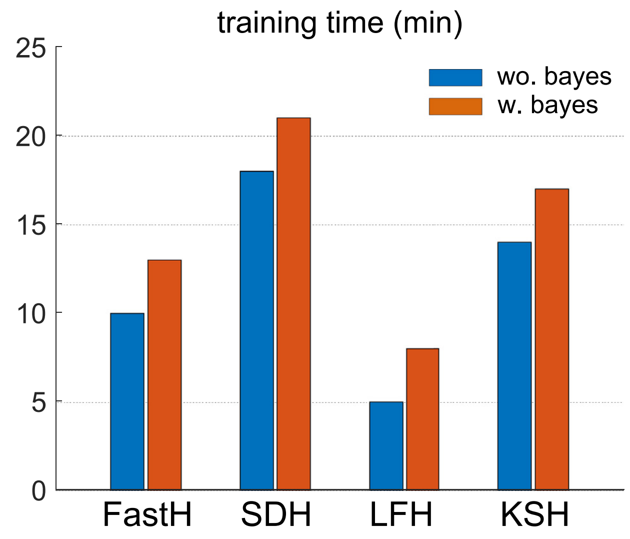


Fig. 8. The training time (min) of hashing models on CIFAR-10 dataset (24-bits code).

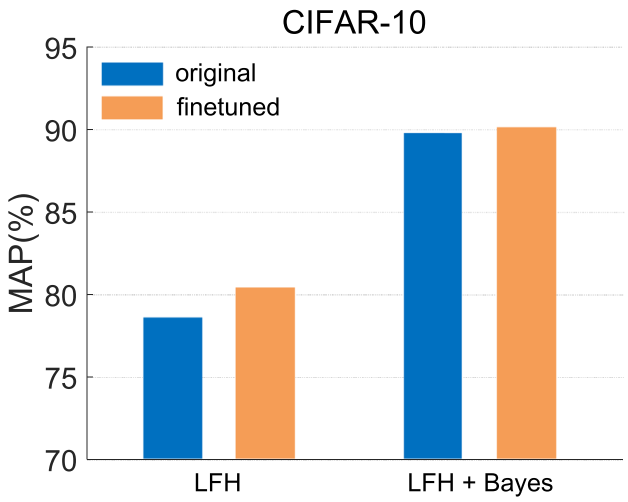


Fig. 9. The effect of finetuning on CIFAR-10 dataset (24-bits code).

improve the MAP score of FashH by 10.0% from 52.4 to 62.4% at 512 bits coding with no occlusion, while at the extreme case of 80 × 80 in pixel occlusion and 512 bits coding, the relative performance improvement yielded by our method is 12.9% (from 46.7 to 59.6%). The results also show that the performance of short hashing codes on this dataset is far from being satisfied (even using our Bayesian denoising hashing layer) for this dataset. This indicate that the length of hashing codes may depend on the complexity of the content of the underlying dataset, and in order to encode a big dataset of many categories, long hashing codes are better.

Fig. 11 gives some illustration of the retrieval results. We argue that for real-world images like those in the ImageNet, there inevitably contains complicated background or other random structural distortion that deteriorates the retrieval performance (cf., Fig. 11). This highlights the need for post-processing methods like ours to further handle the 'raw' hash code. Note that it seems that the DAE does not work well in this case, possibly due to the fact that the small Gaussian noise adopted in DAE is not consistent with the patterns that disturb the hash codes.

3.5. Discussion

Our experiments reported above clearly show that current deep hashing methods are sensitive to the structural noisy signal contained in the input image. Ensemble methods like Fast Hash [37] are seen to alleviate noise on input features by constructing robust hash function with a set of classifiers, which work with each other to improve noise resistance. However, as we see in Table 1, this is still not enough for real-world applications.

One natural question is, since both previous learning to hash methods and our Bayesian denoising hashing method assume a supervised learning setting to identify behaviorally relevant hash bits, why our method could improve the performance of previous state-of-the-art hashing methods even further? In our opinion, there are

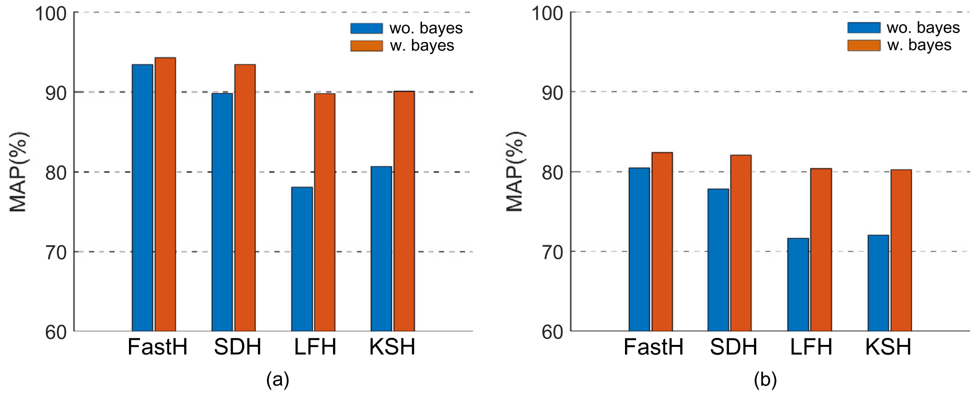


Fig. 10. The effect of network structure: (a) Wide ResNet network [73], (b) CNN-F network [74].

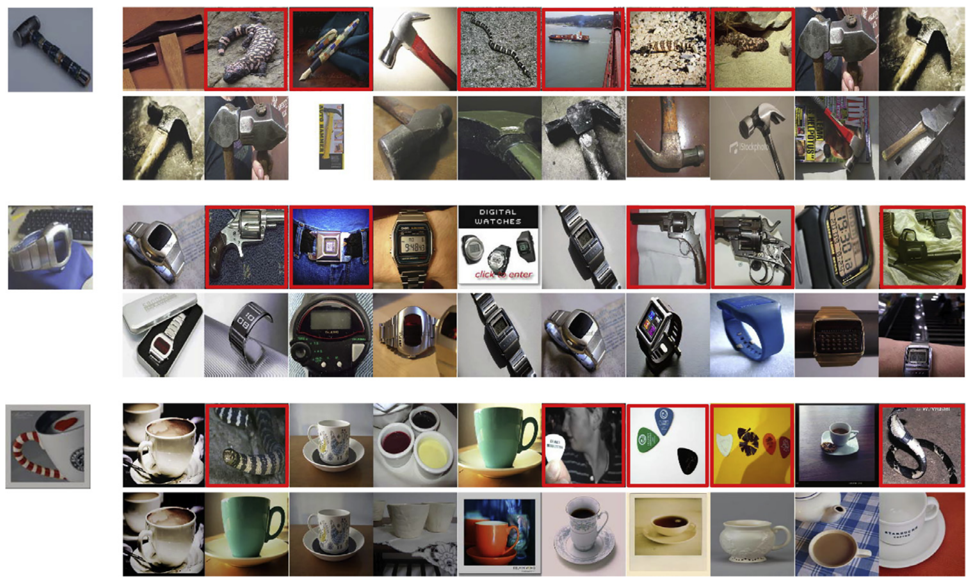


Fig. 11. Top10 retrieved images on ImageNet of FastH and FastH + Bayes under 2048 bits. For each query image (the first column), images in the first row are the top 10 results retrieved with FastH method, while the images below are results by using FastH+Bayes. The red square indicates the false positive. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

two factors that play an important role in explaining this. The first one is due to the variational inference mechanism. One can see from Eq. (9) that the estimated 'cleaned' code $\bar{z}_i$ is computed by combining the information from the original hash code (i.e., x_i) and its pairwise constraints z_j, $j \in N_i$, while W can be interpreted as combination coefficients. Hence this is actually a voting strategy which makes a tradeoff between preserving the original information and correcting it based on the supervised pairwise constraints. Moreover, Eq. (21) ensures that the risk of "over-correcting", i.e., flipping the hash code too much, is very low. The second source of our robustness actually comes from the previous hashing layer it aims to denoise - note that this lower hashing layer works directly on the real-valued input (i.e., CNN features) by performing a non-linear transformation on it, so that the influence of unknown random disturbance on the raw data is partially 'blocked' or bounded by just "bit flipping" (from 0 to 1 or from 1 to 0). This helps to alleviate the burden faced by our method.

4. Conclusion

The robustness of hash coding against structural noise is an important yet seldom-studied problem. General hashing methods are devoted to dealing with real-valued data, and therefore are sensitive to the noise contained in data, which would possibly result in heavy performance deterioration. In this paper we present a Bayesian denoising hashing within the variational Bayes framework. Our method serves as a post-processing denoising layer by exploiting supervised constraints of data, which corrects the error hash bits learned from other hashing algorithms. Experimental results demonstrate its performance, scalability and efficiency on several challenging image retrieval tasks.

However, the performance of this method relies on the input CNN features which would possibly limit its applicability. In future work, we will focus on learning a Bayesian deep hashing model in an end-to-end way, and thus make it more applicable for complex datasets.

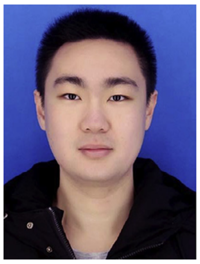
Acknowledgments

This work is partially supported by National Natural Science Foundation of China (61672280, 61373060 and 61732006), the National Key R&D Program of China (No. 2017YFB0802300), Jiangsu 333 Project (BRA2017377), the Pre-research fund of EDD, Qing Lan Project and the Funding of Jiangsu Innovation Program for Graduate Education (KYLX15_0320).

References

[1] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Trans. Pattern Anal. Mach. Intell. 22 (12) (2000) 1349–1380.
[2] W. Zhou, H. Li, R. Hong, Y. Lu, Q. Tian, Bsift: toward data-independent codebook for large scale image search, IEEE Trans. Image Process. 24 (3) (2015) 967–979.
[3] X. Yang, X. Qian, T. Mei, Learning salient visual word for scalable mobile image retrieval, Pattern Recognit. 48 (10) (2015) 3093–3101.
[4] X. Yang, X. Qian, Y. Xue, Scalable mobile image retrieval by exploring contextual saliency, IEEE Trans. Image Process. 24 (6) (2015) 1709.
[5] W. Zhou, H. Li, Y. Lu, M. Wang, Q. Tian, Visual word expansion and bsift verification for large-scale image search, Multimedia Syst. 21 (3) (2015) 245–254.
[6] S. Zhang, Q. Tian, G. Hua, Q. Huang, W. Gao, Generating descriptive visual words and visual phrases for large-scale image applications, IEEE Trans. Image Process. 20 (9) (2011) 2664–2677.
[7] X. Qian, H. Wang, Y. Zhao, X. Hou, R. Hong, Y.Y. Tang, Y.Y. Tang, Image location inference by multisaliency enhancement, IEEE Trans. Multimedia 19 (4) (2017) 813–821.
[8] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Neural Information Processing Systems, 25, 2012, pp. 1097–1105.
[9] S. Pang, J. Zhu, J. Wang, V. Ordonez, J. Xue, Building discriminative cnn image representations for object retrieval using the replicator equation, Pattern Recognit. (2018).
[10] M. Sajjad, S. Khan, T. Hussain, K. Muhammad, A.K. Sangaiah, A. Castiglione, C. Esposito, S.W. Baik, Cnn-based anti-spoofing two-tier multi-factor authentication system, Pattern Recognit. Lett. (2018).
[11] C. Szegedy, A. Toshev, D. Erhan, Deep neural networks for object detection, Proc. NIPS (2013) 2553–2561.
[12] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, Comput. Sci. 79 (10) (2015) 1337–1342.
[13] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: Proc. ECCV, 2014, pp. 584–599.
[14] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, Cnn features off-the-shelf: An astounding baseline for recognition, in: Proc. CVPR Workshops, 2014, pp. 512–519.
[15] A. Babenko, V. Lempitsky, Aggregating deep convolutional features for image retrieval, Comput. Sci. (2015).
[16] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of cnn activations, arXiv preprintarXiv:1511.05879 (2015).
[17] Y.H. Ng, F. Yang, L.S. Davis, Exploiting local features from deep networks for image retrieval, Computer Science (2015) 53–61.
[18] Y. Zhang, H. Lu, L. Zhang, X. Ruan, S. Sakai, Video anomaly detection based on locality sensitive hashing filters, Pattern Recognit. 59 (2016) 302–311.

[19] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, H.T. Shen, Robust discrete code modeling for supervised hashing, Pattern Recognit. 75 (2018) 128–135.
[20] J. Song, L. Gao, L. Liu, X. Zhu, N. Sebe, Quantization-based hashing: a general framework for scalable image and video retrieval, Pattern Recognit. 75 (2018) 175–187.
[21] J. Tang, Z. Li, X. Zhu, Supervised deep hashing for scalable face image retrieval, Pattern Recognit. 75 (2018) 25–32.
[22] V.E. Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, Deep hashing for compact binary codes learning, in: Proc. CVPR, 2015.
[23] P. Zhang, W. Zhang, W.-J. Li, M. Guo, Supervised hashing with latent factor models, in: ACM SIGIR, ACM, 2014, pp. 173–182.
[24] W.-J. Li, S. Wang, W.-C. Kang, Feature learning based deep supervised hashing with pairwise labels, in: IJCAI, Proc. AAAI Press, 2016, pp. 1711–1717.
[25] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: Proc. CVPR, 2015.
[26] J. Zhang, Y. Peng, Ssdh: semi-supervised deep hashing for large scale image retrieval, IEEE Trans. Circuits Syst. Video Technol. (2017).
[27] J. Zhang, Y. Peng, Query-adaptive image retrieval by deep weighted hashing, IEEE Trans. Multimedia (2018).
[28] J. Zhang, Y. Peng, M. Yuan, Unsupervised generative adversarial cross-modal hashing, arXiv preprintarXiv:1712.00358 (2017).
[29] J. Zhang, Y. Peng, M. Yuan, Sch-gan: semi-supervised cross-modal hashing by generative adversarial network, arXiv preprintarXiv:1802.02488 (2018a).
[30] J. Zhang, Y. Peng, Z. Ye, Deep reinforcement learning for image hashing, arXiv preprintarXiv:1802.02904 (2018b).
[31] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, Web scale photo hash clustering on a single machine, in: Proc. CVPR, 2015, pp. 19–27.
[32] R. Salakhutdinov, G. Hinton, Semantic hashing, Int. J. Approximate Reasoning 50 (7) (2009) 969–978.
[33] F. Zhu, X. Kong, L. Zheng, H. Fu, Q. Tian, Part-based deep hashing for large-scale person re-identification, IEEE Trans. Image Process. 26 (10) (2017) 4806–4817.
[34] Q. Li, Z. Sun, R. He, T. Tan, Deep supervised discrete hashing, in: Advances in Neural Information Processing Systems, 2017, pp. 2482–2491.
[35] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised hashing with kernels, in: Proc. CVPR, 2012, pp. 2074–2081.
[36] L. Fan, Supervised binary hash code learning with jensen shannon divergence, in: Proc. ICCV, 2013, pp. 2616–2623.
[37] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, Fast supervised hashing with decision trees for high-dimensional data, in: Proc. CVPR, 2014, pp. 1971–1978.
[38] F. Shen, C. Shen, W. Liu, H.T. Shen, Supervised discrete hashing, in: Proc. CVPR, 2015, pp. 37–45.
[39] K. Ding, C. Huo, B. Fan, C. Pan, knn hashing with factorized neighborhood representation, in: Proc. ICCV, 2015, pp. 1098–1106.
[40] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing., in: Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December, 2008, pp. 1753–1760.
[41] W. Kong, W.J. Li, Isotropic hashing, in: Proc. NIPS, 2, 2012, pp. 1646–1654.
[42] W. Liu, C. Mu, S. Kumar, S.F. Chang, Discrete graph hashing, in: Proc. NIPS, 4, 2014, pp. 3419–3427.
[43] F. Shen, W. Liu, S. Zhang, Y. Yang, Learning binary codes for maximum inner product search, in: Proc. ICCV, 2015.
[44] J. Wang, S. Kumar, S.F. Chang, Semi-supervised hashing for scalable image retrieval, in: Proc. CVPR, 2010, pp. 3424–3431.
[45] S. Kim, S. Choi, Semi-supervised discriminant hashing, in: Proc. ICDM, 2011, pp. 1122–1127.
[46] C.L. Zitnick, T. Kanade, A cooperative algorithm for stereo matching and occlusion detection, IEEE Trans. Pattern Anal. Mach. Intell. 22 (7) (2000) 675–684.
[47] A. Ayvaci, M. Raptis, S. Soatto, Occlusion detection and motion estimation with convex optimization, in: Proc. NIPS, 2010, pp. 100–108.
[48] X. Mei, H. Ling, Y. Wu, E. Blasch, L. Bai, Minimum error bounded efficient ? 1 tracker with occlusion detection, in: Proc. CVPR, 2011, pp. 1257–1264.
[49] C. Steger, Occlusion, clutter, and illumination invariant object recognition, Int. Arch. Photogrammetry Remote Sens. Spat. Inf. Sci. 34 (3/A) (2002) 345–350.
[50] S. Chambon, A. Crouzil, Similarity measures for image matching despite occlusions in stereo vision, Pattern Recognit. 44 (9) (2011) 2063–2075.
[51] S. Park, H. Lee, J.-H. Yoo, G. Kim, S. Kim, Partially occluded facial image retrieval based on a similarity measurement, Math. Probl. Eng. 2015 (2015).
[52] Y. Tian, P. Luo, X. Wang, X. Tang, Deep learning strong parts for pedestrian detection, in: Proc. ICCV, 2015, pp. 1904–1912.
[53] H. Fu, C. Wang, D. Tao, M.J. Black, Occlusion Boundary Detection Via Deep Exploration of Context, Proc. CVPR, IEEE, 2016, pp. 241–250.
[54] N. Kwak, Principal component analysis based on l1-norm maximization, IEEE Trans. Pattern Anal. Mach. Intell. 30 (9) (2008) 1672–1680.
[55] Q. Zhao, D. Meng, Z. Xu, W. Zuo, L. Zhang, Robust principal component analysis with complex noise, in: Proc. ICML, 2014, pp. 55–63.
[56] D. Wang, X. Tan, Robust distance metric learning in the presence of label noise, in: Proc. AAAI, 2014.
[57] H. Shen, J.Z. Huang, Sparse principal component analysis via regularized low rank matrix approximation, J. Multivar. Anal. 99 (6) (2008) 1015–1034.
[58] I.M. Johnstone, A.Y. Lu, Sparse principal components analysis, arXiv preprintarXiv:0901.4392 (2009).
[59] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM 58 (3) (2011) 11.
[60] D. Wang, X. Tan, Bayesian neighborhood component analysis, IEEE Trans. Neural Netw. Learn. Syst. PP (99) (2017) 1–12.
[61] D. Wang, X. Tan, Robust distance metric learning via bayesian inference, IEEE Trans. Image Process. 27 (3) (2018) 1542–1553.
[62] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th Proc. ICML, ACM, 2008, pp. 1096–1103.
[63] Y. Tang, R. Salakhutdinov, G. Hinton, Robust boltzmann machines for recognition and denoising, in: Proc. CVPR, IEEE, 2012, pp. 2264–2271.
[64] X.-J. Mao, C. Shen, Y.-B. Yang, Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections, arXiv preprint (2016).
[65] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: residual learning of deep cnn for image denoising, IEEE Trans. Image Process. (2017).
[66] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, arXiv preprintarXiv:1512.03385 (2015).
[67] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul, An introduction to variational methods for graphical models, Mach. Learn. 37 (2) (1999) 183–233.
[68] T.S. Jaakkola, M.I. Jordan, Bayesian parameter estimation via variational methods, Stat. Comput. 10 (1) (2000) 25–37.
[69] A. Coates, H. Lee, A.Y. Ng, An analysis of single-layer networks in unsupervised feature learning, Ann Arbor 1001 (2010) 48109.
[70] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[71] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Proc. CVPR, IEEE, 2009, pp. 248–255.
[72] D. Wang, X. Tan, Unsupervised feature learning with c-svddnet, Pattern Recognit. 60 (2016) 473–485.
[73] S. Zagoruyko, N. Komodakis, Wide residual networks, in: Proc. BMVC, 2016.
[74] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving into convolutional nets, arXiv preprintarXiv:1405.3531 (2014).



Dong Wang is currently working toward the Ph.D. degree at the Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include robust learning with label noise, Bayesian learning, metric learning and feature learning.

Ge Song is currently working toward the Ph.D. degree at the Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include image retrieval, hashing learning and deep learning.

Xiaoyang Tan is currently a Professor of the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. From 2006 to 2007, he was a Postdoctoral Researcher with the LEAR Team, INRIAR Rhone-Alpes, Grenoble, France. He has authored or coauthored over 50 conference and journal papers. His research interests include deep learning, reinforcement learning, and Bayesian learning. He and his colleagues were awarded the *IEEE Signal Processing Society Best Paper* in 2015.