Unsupervised Deep Generative Hashing

Yuming Shen[1]
yuming.shen@uea.ac.uk
Li Liu[1,2]
li.liu@malongtech.cn
Ling Shao[1]
ling.shao@ieee.org

[1] School of Computing Sciences
University of East Anglia
Norwich, UK

[2] Malong Technologies Co., Ltd.
Shenzhen, China

Abstract

Hashing is regarded as an efficient approach for image retrieval and many other big-data applications. Recently, deep learning frameworks are adopted for image hashing, suggesting an alternative way to formulate the encoding function other than the conventional projections. However, existing deep learning based unsupervised hashing techniques still cannot produce leading performance compared with the non-deep methods, as it is hard to unveil the intrinsic structure of the whole sample space in the framework of mini-batch Stochastic Gradient Descent (SGD). To tackle this problem, in this paper, we propose a novel unsupervised deep hashing model, named Deep Variational Binaries (DVB). The conditional auto-encoding variational Bayesian networks are introduced in this work as the generative model to exploit the feature space structure of the training data using the latent variables. Integrating the probabilistic inference process with hashing objectives, the proposed DVB model estimates the statistics of data representations, and thus produces compact binary codes. Experimental results on three benchmark datasets, *i.e.,* CIFAR-10, SUN-397 and NUS-WIDE, demonstrate that DVB outperforms state-of-the-art unsupervised hashing methods with significant margins.

1 Introduction

Embedding high-dimensional data representations to low dimensional binary codes, hashing algorithms arouse wide research attention in computer vision, machine learning and data mining. Considering the low computational cost of approximate nearest neighbour search in the Hamming space, hashing techniques deliver more effective and efficient large-scale data retrieval than real-valued embeddings. Hashing methods can be typically categorized as either supervised or unsupervised hashing, while this paper focuses on the latter.

Supervised hashing [10, 19, 26, 27, 34, 39, 42] utilises data labels or pair-wise similarities as supervision during parameter optimization. It attains relatively better retrieval performance than the unsupervised models as the conventional evaluation measurements of data retrieval are highly related to the labels. However, due to the cost of manual annotation and tagging, supervised hashing is not always appreciated and demanded. On the other hand, unsupervised hashing [9, 11, 12, 17, 24, 25, 29, 30, 37, 38, 43, 47] learns the binary encoding function based on data representations and require no label information, which eases the task of data retrieval where human annotations are not available.

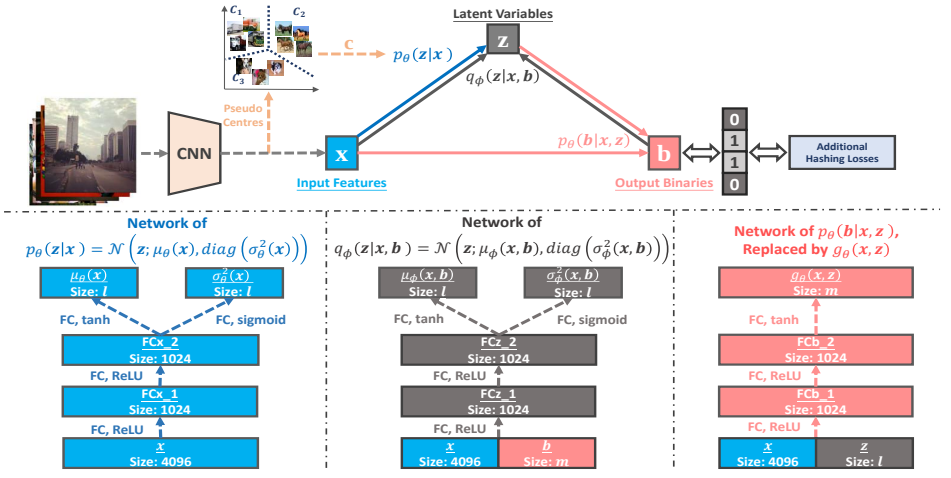


Figure 1: Illustration of DVB as a graphical model. The arrowed full lines with different colours indicate different probability models implemented with deep neural networks. In particular, $p_\theta(\mathbf{z}|\mathbf{x})$ in blue acts as the (conditional) prior of the latent variables $\mathbf{z}$; $p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})$ refers to the generation network for $\mathbf{b}$; $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ is the variational posterior of $\mathbf{z}$. The component computing data centres assigns a low-dimensional pseudo centre $\mathbf{c}$ to each data point $\mathbf{x}$ using some dimensionality reduction and clustering methods. Implementation details are given in Section 2.

Existing research interests on unsupervised hashing involve various strategies to formulate the encoding functions. For instance, Gong *et al*. propose the Iterative Quantization (ITQ) [9], aiming at minimizing quantization error to produce binary representations. Spectral Hashing (SH) developed by Weiss *et al*. [43] learns the hash function by preserving the balanced and uncorrelated constraints of the learnt codes. Liu *et al*. employ unsupervised graph hashing with discrete constraints, known as Discrete Graph Hashing (DGH) [32]. Mathematically profound as these works are, the performance of the shallow unsupervised hashing on similarity retrieval is still far from satisfying. This is possibly due to the fact that the simple encoding functions, *e.g*., linear projections, in these works are not capable to handle complex data representations, and therefore the generated codes are suspected to be less informative.

Recently, deep learning is introduced into image hashing, suggesting an alternative manner of formulating the binary encoding function. Although supervised deep hashing has been proved to be successful [3, 21, 28, 44, 48], existing works on unsupervised deep hashing [4, 8, 22, 23] are yet suboptimal. Different from the conventional shallow methods mentioned above [9, 31, 32], unsupervised deep hashing models follow the mini-batch Stochastic Gradient Decent (SGD) routine for parameter optimization. Consequently, providing no label information, the intrinsic structure and similarities of the whole sample space can be skewed within training batches by these models.

Driven by the issues discussed above, a novel deep unsupervised hashing algorithm is proposed which utilises the structural statistics of the whole training data to produce reliable binary codes. The auto-encoding variational algorithms [16] have shown great potential in several applications [20, 46]. The recent Conditional Variational Auto-Encoding (CVAE)

networks [41] provide an illustrative way to build a deep generative model for structured outputs, by which we are inspired to establish our deep hashing model, named as Deep Variational Binaries (DVB). In particular, the latent variables of the variational Bayesian networks [16] are leveraged to approximate the representation of the pre-computed pseudo clustering centre that each data point belongs to. Thus the binary codes can be learnt as informative as the input features by maximizing the conditional variational lower bound of the our learning objective. It is worth noticing that we are not using the quantized latent variables as binary representations. Instead, the latent variables are treated as auxiliary data to generate the conditional outputs as hashed codes. By the time of writing, we are aware that Chaidaroon *et al*. [5] propose a variational binary encoder for text hashing. However, [5] is not suitable for image encoding since it takes discrete word count vectors as input, while images would have longer and more complex representations.

The contribution of this paper can be summarized as: **a)** to the best of our knowledge, DVB is the first unsupervised deep hashing work in the framework of variational inference suitable for image retrieval; **b)** the proposed deep hashing functions are optimized efficiently, requiring no alternating training routine; **c)** DVB outperforms state-of-the-art unsupervised hashing methods by significant margins in image retrieval on three benchmarked datasets, *i.e.,* CIFAR-10, SUN-397 and NUS-WIDE.

2 Deep Variational Binaries

This work addresses the problem of data retrieval with an unsupervised hashing procedure. Given a data collection $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{d\times N}$ consisting N data points with d-dimensional real-valued representations, the DVB model learns an encoding function $f(\cdot)$, parametrized by θ, so that each data point can be represented as

$$\mathbf{b}_i = \mathtt{sign}\left(f(\mathbf{x}_i;\theta)\right) \in \{-1,1\}^m. \tag{1}$$

Here m indicates the encoding length and $\mathtt{sign}(\cdot)$ refers to the sign function for quantization. In the following description, index i will be omitted when it clearly refers to a single data point. In this section, we firstly explain the way to empirically exploit the intrinsic structure of the training set by introducing a set of latent variables $\mathbf{z}$ and then, the encoding function $f(\cdot)$ is formulated by a Monte Carlo sampling procedure for out-of-sample extension.

2.1 The Variational Model

As shown in Figure 1, the DVB framework involves three types of variables, *i.e.,* the data representations $\mathbf{x} \in \mathbb{R}^d$, the output codes $\mathbf{b} \in \{-1,1\}^m$ and the latent representations $\mathbf{z} \in \mathbb{R}^l$ as auxiliary variables, where l denotes the dimensionality of the latent space. The variables in DVB formulate three probabilistic models, *i.e.,* the conditional prior $p_\theta(\mathbf{z}|\mathbf{x})$, the variational posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ and the generation network $p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})$. Following Kingma *et al*. [16], the probability models here are implemented using deep neural networks, parametrized by θ or ϕ. We consider the prototype of learning objective maximizing the log-likelihood $\log p_\theta(\mathbf{b}|\mathbf{x})$ for each training data point by approximating the true posterior $p_\theta(\mathbf{z}|\mathbf{x},\mathbf{b})$ using $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$. Starting with the K-L divergence between $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ and $p_\theta(\mathbf{z}|\mathbf{x},\mathbf{b})$:

$$KL(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) \parallel p_\theta(\mathbf{z}|\mathbf{x},\mathbf{b})) = \int q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}{p_\theta(\mathbf{z},\mathbf{b}|\mathbf{x})} d\mathbf{z} + \log p_\theta(\mathbf{b}|\mathbf{x}), \tag{2}$$

the likelihood of $\mathbf{b}$ can be written as

$$\begin{aligned}\log p_\theta(\mathbf{b}|\mathbf{x}) &= KL(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) \parallel p_\theta(\mathbf{z}|\mathbf{x},\mathbf{b})) - \int q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}{p_\theta(\mathbf{z},\mathbf{b}|\mathbf{x})} d\mathbf{z} \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}\left[\log p_\theta(\mathbf{b},\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})\right].\end{aligned} \tag{3}$$

Here the expectation term $\mathbb{E}[\cdot]$ becomes the prototype of the learning objective of DVB. Considering the deep neural networks mentioned above, we follow a similar way of [41] to factorize the lower-bound, and thus we have

$$\begin{aligned}-\log p_\theta(\mathbf{b}|\mathbf{x}) \leq \mathcal{L} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}\left[\log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) - \log p_\theta(\mathbf{b},\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}\left[\log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) - \log p_\theta(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})\right] \\ &= KL\left(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) || p_\theta(\mathbf{z}|\mathbf{x})\right) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}\left[\log p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})\right].\end{aligned} \tag{4}$$

We denote $\mathcal{L}$ as the numerical **inverse** of the lower-bound to $\log p_\theta(\mathbf{b}|\mathbf{x})$ for the ease of description in the rest of this paper. Therefore DVB performs SGD to **minimize** $\mathcal{L}$.

As image data are usually presented in high-dimensional representations, directly reconstructing $\mathbf{x}$ from $\mathbf{z}$ as [5, 16] is not optimal and could induce redundant noise to the training procedure. In DVB, $\mathbf{z}$ act as auxiliary variables encoding latent information through the conditional network $p_\theta(\mathbf{z}|\mathbf{x})$. By reducing the divergence between the posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ and $p_\theta(\mathbf{z}|\mathbf{x})$, the generated binaries $\mathbf{b}$ are supposed to have similar semantics to the original feature $\mathbf{x}$ in reconstructing $\mathbf{z}$. To solve the intractability of the posterior $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$, the inference network is built using the reparameterizaion trick in [16] with a Gaussian distribution so that

$$q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b}) = \mathcal{N}\left(\mathbf{z};\ \mu_\phi(\mathbf{x},\mathbf{b}), \mathtt{diag}\left(\sigma^2_\phi(\mathbf{x},\mathbf{b})\right)\right). \tag{5}$$

Note that all $\mu_\cdot(\cdot)$ and $\sigma_\cdot(\cdot)$ can be implemented with multi-layer neural networks, which is provided in Figure 1. A similar trick is also performed on $p_\theta(\mathbf{z}|\mathbf{x})$ as follows

$$p_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z};\ \mu_\theta(\mathbf{x}), \mathtt{diag}\left(\sigma^2_\theta(\mathbf{x})\right)\right). \tag{6}$$

Although the continues distributions $p_\theta(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ can be reparameterized, it is still hard to model $p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})$ because $\mathbf{b}$ needs to be discrete and there is no additional supervision available. Hence, the log-likelihood $\log p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})$ is replaced by a serious of deep hashing learning objectives $\mathcal{H}(\cdot)$, *i.e.,* $-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}[\log p_\theta(\mathbf{b}|\mathbf{x},\mathbf{z})] \longrightarrow \mathcal{H}(g_\theta(\mathbf{x},\mathbf{z}))$. Here $g_\theta(\cdot)$ refers to the deep neural network to generate $\mathbf{b}$ so that $\mathbf{b} = \mathtt{sign}(g_\theta(\mathbf{x},\mathbf{z}))$.

Exploiting the intrinsic data structure. In addition to the lower-bound mentioned above, we consider utilising the statistical information on the whole training set for better performance. Inspired by [31, 32], a small set of K anchor points $\{\mathbf{c}^j\}_{j=1}^K \in \mathbb{R}^{l\times K}$ are computed before the SGD training starts, which is also shown in Figure 1. Each anchor point refers to a pseudo clustering centre of the training data. Then each data point $\mathbf{x}_i$ is assigned with a clustering centre by nearest neighbour search, *i.e.,* $\{\mathbf{x}_i, \mathbf{c}_i\}$. In practice, this is achieved by successively performing dimension reduction and clustering on the training set. Different from [31, 32], we are not building the anchor graph on the whole dataset since this is not practical for mini-batch SGD. Instead, the latent variable $\mathbf{z}$ is used to predict the representation of the corresponding anchor $\mathbf{c}$ of each $\mathbf{x}$. More precisely, the mean network $\mu_\theta(\mathbf{x})$ of $p_\theta(\mathbf{z}|\mathbf{x})$ is related to $\mathbf{c}$, formulating an additional $l2$ loss term, which particularly requires $\mathbf{z}$ have the same dimensionality as $\mathbf{c}$. This procedure intuitively endows the conditional network $p_\theta(\mathbf{z}|\mathbf{x})$ with more informative latent semantics. Therefore, the total learning

objective can be written as

$$\widetilde{\mathcal{L}} = KL\left(q_\phi\left(\mathbf{z}|\mathbf{x},\mathbf{b}\right)||p_\theta\left(\mathbf{z}|\mathbf{x}\right)\right) + \mathcal{H}\left(g_\theta(\mathbf{x},\mathbf{z})\right) + \|\mu_\theta\left(\mathbf{x}\right) - \mathbf{c}\|^2. \tag{7}$$

Note that $\widetilde{\mathcal{L}}$ here can be no longer regarded as the exact lower-bound of $\log p_\theta\left(\mathbf{b}|\mathbf{x}\right)$. This empirical learning objective partially represents the likelihood of $\mathbf{b}$ with more hashing concerns. In the next subsection, the details of the hashing objective term $\mathcal{H}\left(g_\theta(\mathbf{x},\mathbf{z})\right)$ is discussed.

2.2 Hashing Objectives

The hashing objective $\mathcal{H}\left(\cdot\right)$ in Equation (7) in replacement of $-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})}\left[\log p_\theta\left(\mathbf{b}|\mathbf{x},\mathbf{z}\right)\right]$ is formulated by several unsupervised loss components to regular the output of the proposed hashing model. Since DVB is trained using mini-batch SGD, the losses need to be able to back-propagate. Inspired by several unsupervised deep hashing works [8, 22, 23], we formulate the following hashing losses to construct $\mathcal{H}\left(\cdot\right)$ within a batch of data points $\mathbf{X}_B = \{\mathbf{x}_i\}_{i=1}^{N_B}$ and sampled latent variables $\mathbf{Z}_B = \{\mathbf{z}_i\}_{i=1}^{N_B}$, where N_B is the batch size.

Quantization Penalty. As DVB produces binary codes, the output bits of $g_\theta\left(\cdot\right)$ need to be close to either 1 or -1. This minimizes the numerical gap between the network output and the quantized product of the $\texttt{sign}\left(\cdot\right)$ function. The quantization loss can thus be written with a Frobenius norm as follows

$$\mathcal{H}_1 = \|g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right) - \texttt{sign}\left(g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right)\right)\|_{\mathrm{F}}^2. \tag{8}$$

The quantization losses are widely adopted in several hashing works [8, 22, 23, 48] with different formulations. In our experiments, we find the Frobenius norm works best for DVB with a $\texttt{tanh}$ activation on the top layer of $g_\theta\left(\cdot\right)$.

Bit Decorrelation. The encoded binaries in hashing algorithms are in general short in length. To make the produced code representative, it is necessary to decorrelate each bit and balance the quantity of 1 and -1 in a code vector. To this end, the second component of $\mathcal{H}\left(\cdot\right)$ is derived as

$$\mathcal{H}_2 = \|g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right)^\mathsf{T} g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right) - \mathbf{I}\|_{\mathrm{F}}^2, \tag{9}$$

where $\mathbf{I}$ refers to the identity matrix and both $\mathbf{X}_B$ and $\mathbf{Z}_B$ are row-ordered matrices. Equation (9) suggests an indirect way to enrich the information encoded in the binary codes by balancing the output bits.

In-Batch Similarity. For unsupervised hashing, it is usually in demand to closely encode data samples that have similar representations into the Hamming space. Inspired by [2, 13], the in-batch Laplacian graph is introduced to build the last term of $\mathcal{H}\left(\cdot\right)$. To do this, an in-batch Laplacian matrix is defined by $\mathbf{S} = \texttt{diag}\left(\mathbf{A}\mathbb{1}\right) - \mathbf{A}$. Here $\mathbf{A}$ is an $N_B \times N_B$ distance matrix of which each entrance $\mathbf{A}_{ij}$ is computed by $\mathbf{A}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}$, where t is a small-valued hyper-parameter. A trace-like learning objective for in-batch similarity can be written as

$$\mathcal{H}_3 = -\texttt{trace}\left(g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right)^\mathsf{T}\mathbf{S}\, g_\theta\left(\mathbf{X}_B,\mathbf{Z}_B\right)\right). \tag{10}$$

$\mathcal{H}_3$ functionally works similarly to the pre-computed low-dimensional clustering centres $\mathbf{c}$ in preserving the unlabelled data similarities. However, $\mathcal{H}_3$ focuses on regulating $\mathbf{b}$ within a batch while $\mathbf{c}$ provides support to form the latent space $\mathbf{z}$ on the whole training set.

Therefore, $\mathcal{H}$ in Equation (7) can be formulated by a weighted combination of $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, *i.e.*, $\mathcal{H}\left(g_\theta(\mathbf{x},\mathbf{z})\right) = \alpha_1\mathcal{H}_1 + \alpha_2\mathcal{H}_2 + \alpha_3\mathcal{H}_3$, where α_1, α_2 and α_3 are treated as hyper-parameters.

Algorithm 1: Parameter Learning of DVB

Input: A dataset with representations $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{d\times N}$
Output: network parameters θ and ϕ
Perform dimension reduction and clustering on the dataset to have $\{\mathbf{c}\}$
repeat
  Get a random mini-batch $\mathbf{X}_B$ from $\mathbf{X}$
  for *each* $\mathbf{x}_i$ *in* $\mathbf{X}_B$ **do**
    Relate $\mathbf{x}_i$ with a closest clustering centre representation $\mathbf{c}_i$
    Sample $\mathbf{z}_i \sim p_\theta(\mathbf{z}|\mathbf{x}_i)$ following [16]
    $\mathbf{b}_i = \mathtt{sign}(g_\theta(\mathbf{x}_i, \mathbf{z}_i))$
  end
  $\widetilde{\mathcal{L}_B} \leftarrow$ Equation (11)
  $(\theta,\phi)^{\text{new}} \leftarrow (\theta,\phi) - \Gamma\left(\nabla_\theta \widetilde{\mathcal{L}_B}, \nabla_\phi \widetilde{\mathcal{L}_B}\right)$ by back-propagation
until *convergence*;

2.3 Optimization

By introducing the hashing losses discussed in Subsection 2.2 into DVB, the overall learning objective $\widetilde{\mathcal{L}_B}$ on a mini-batch $\mathbf{X}_B$ can be written as follows

$$\widetilde{\mathcal{L}_B} = \sum_{i=1}^{N_B} \left(KL\left(q_\phi(\mathbf{z}_i|\mathbf{x}_i, \mathbf{b}_i) || p_\theta(\mathbf{z}_i|\mathbf{x}_i)\right) + \|\mu_\theta(\mathbf{x}_i) - \mathbf{c}_i\|^2 \right) + \alpha_1\mathcal{H}_1 + \alpha_2\mathcal{H}_2 + \alpha_3\mathcal{H}_3. \quad (11)$$

The SGD training procedure of DVB is illustrated in Algorithm 1. For each data point $\mathbf{x}_i$ within a batch $\mathbf{X}_B$, a latent representation $\mathbf{z}_i$ is obtained by sampling the conditional distribution $p_\theta(\cdot|\mathbf{x}_i)$ and an estimated binary vector $\mathbf{b}_i$ can be calculated by $\mathbf{b}_i = \mathtt{sign}(g_\theta(\mathbf{x}_i, \mathbf{z}_i))$ to further compute $\widetilde{\mathcal{L}_B}$. The parameters (θ,ϕ) are updated following the mini-batch SGD. $\Gamma(\cdot)$ here refers to an adaptive gradient scaler, which is the Adam optimizer [15] in this paper with a starting learning rate of 10^{-4}.

2.4 Out-of-sample extension

Once the set of parameters θ is trained, the proposed DVB model is able to encode data out of the training set. Given a query data $\mathbf{x}^q$, the corresponding binary code $\mathbf{b}^q$ can be obtained by a Monte Carlo (MC) sampling procedure defined as

$$\begin{aligned} f(\mathbf{x}^q;\theta) &= \frac{1}{L}\sum_{s=1}^{L} g_\theta\left(\mathbf{x}^q, \mathbf{z}_{(s)}\right), \quad \mathbf{z}_{(s)} \sim p_\theta(\mathbf{z}|\mathbf{x}^q), \\ \mathbf{b}^q &= \mathtt{sign}\left(f(\mathbf{x}^q;\theta)\right), \end{aligned} \quad (12)$$

which simulates the sampling trick described in [16, 41]. In the experiments of this work, L is fixed to 10 for best performance via cross-validation.

3 Experiments

The extensive experiments of DVB are conducted on three benchmark image datasets, *i.e.,* CIFAR-10 [18], SUN-397 [45] and NUS-WIDE [7] for image retrieval. We firstly introduce

Method	Deep Hashing	CIFAR-10			SUN-397			NUS-WIDE		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
ITQ [9]	✗	0.319	0.334	0.347	0.047	0.070	0.086	0.512	0.526	0.538
SH [43]	✗	0.218	0.198	0.181	0.021	0.033	0.048	0.346	0.358	0.365
SpH [14]	✗	0.229	0.253	0.283	0.032	0.039	0.043	0.418	0.456	0.474
LSH [6]	✗	0.163	0.182	0.232	0.006	0.007	0.011	0.410	0.416	0.439
SKLSH [35]	✗	0.103	0.112	0.114	0.005	0.006	0.008	0.377	0.379	0.388
SELVE [49]	✗	0.309	0.281	0.239	0.049	0.072	0.089	0.467	0.462	0.432
AGH [31]	✗	0.301	0.270	0.238	0.059	0.057	0.062	0.498	0.476	0.471
DGH [32]	✗	0.332	0.354	0.356	0.061	0.074	0.079	0.530	0.527	0.496
DH [23]	✓	0.172	0.176	0.179	0.035	0.047	0.056	0.404	0.467	0.427
DeepBit [22]	✓	0.193	0.216	0.219	0.029	0.058	0.061	0.452	0.463	0.496
UN-BDNN [8]	✓	0.301	0.309	0.312	0.062	0.073	0.088	0.513	0.517	0.547
DVB (proposed)	✓	**0.347**	**0.365**	**0.381**	**0.069**	**0.084**	**0.098**	**0.546**	**0.560**	**0.574**

Table 1: Image retrieval mean-Average Precision (mAP@*all*) on the three datasets with VGG-16 [40] features.

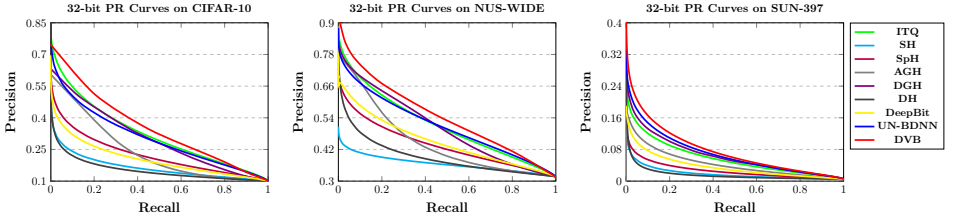
the implementation details, experimental settings and baselines on the three data sets. Then qualitative and quantitative analysis are provided.

Implementation Details. The DVB networks are implemented with the well-known deep learning library TensorFlow [1]. Before being rendered to the DVB networks, a 4096-dimensional deep feature vector of each training image is extracted using the output of the `fc_7` layer of the VGG-16 network [40], pre-trained on ImageNet [36], *i.e.,* $d = 4096$. We follow a similar way presented in [16, 20, 21] to build the deep neural networks $p_\theta(\mathbf{z}|\mathbf{x})$, $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{b})$ and $g_\theta(\mathbf{x},\mathbf{z})$. The detail structures of these networks in DVB are provided Figure 1. The dimensionality of the latent space $\mathbf{z}$ is set to $l = 1024$ via cross-validation. To generate a set of pseudo data centres $\{\mathbf{c}\}$, PCA is performed on the $l2$ normalized training set $\mathbf{X}$ to reduce its dimensionality from 4096 to 1024, followed by a clustering procedure to obtain a set of $\mathbf{c}$. The number of clustering centres K is set according to different datasets. For the rest of the hyper-parameters, t, α_1, α_2 and α_3 are set to 10^{-3}, 0.5, 0.1 and 1 respectively. For all the experiments, the training batch size is fixed to $N_B = 200$.

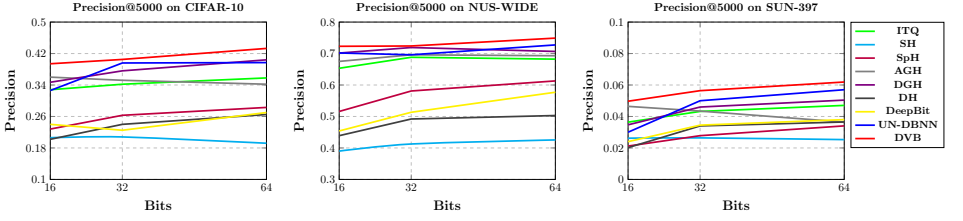
Baselines. Several benchmarked unsupervised hashing methods are involved in the experiments of this paper, including ITQ [9], SH [43], SpH [14], LSH [6], SKLSH [35], SELVE [49], AGH [31] and DGH [32]. Several recent unsupervised deep hashing models are also considered, *i.e.,* DH [23], DeepBit [22] and UN-BDNN [8]. To make a fair comparison between the shallow methods and the deep models, we utilize the VGG-16 [40] features as inputs for all baselines. As a result, the performance figures of the traditional hashing works reported here are slightly higher than those in their original papers, but are still reasonable and illustrative. Three additional baselines are also introduced by removing some components of the learning objective in Equation (11) for ablation study. Particularly, we exclusively omit the $l2$ loss on $\mu_\theta(\cdot)$, $\mathcal{H}_2$ and $\mathcal{H}_3$ to build the three baselines to see the impact of each term in the proposed learning objective.

CIFAR-10 [18]. This dataset consists of 60000 small-size images, subjected to 10 categories. We follow the setting in [32] to randomly select 100 images from each class as the test set, and use the rest 59000 images as the training set and retrieval gallery. K is set to 20 on this dataset by cross-validation.

SUN-397 [45]. A total number of 108754 images are in involved in this dataset with 397 exclusive class labels. For each class, 20 images are randomly selected to form the test set. The reset images are used as training and retrieval candidates. K is set to 500 on this dataset.



(a) 32-bit image retrieval precision-recall curves.



(b) Retrieval Precision@5000 curves for encoding length $m = 16, 32$ and 64.

Figure 2: 32-bit image retrieval Precision-Recall (PR) curves **(a)** and Precision@5000 curves for all bits of DVB and several existing methods **(b)**.

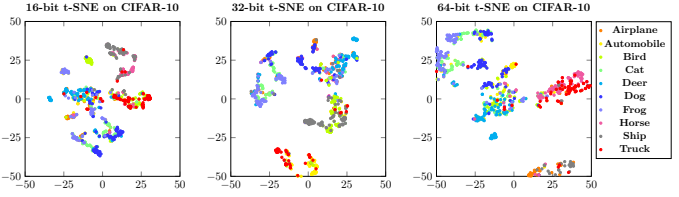
NUS-WIDE [7]. This is a multi-label dataset containing 269648 images. We use a subset of 195834 images from the 21 most frequent topics, from which 100 images for each topic are randomly picked for testing. K is set to 100 on this dataset.

3.1 Quantitative results

The performance of the proposed DVB model is evaluated by conducting image retrieval on the three datasets mentioned above. For experiments on CIFAR-10 [18] and SUN-397 [45], the retrieval candidates having the same label as the query image are marked as the ground-truth relevant data. Since NUS-WIDE [7] is a multi-label dataset, a relevant retrieval candidate is defined as sharing at least one label with the query image, which is a conventional setting in image hashing and retrieval. The code length m is chosen to be 16, 32 and 64.

The image retrieval mean-Average Precision (mAP@*all*) results are provided in Table 1, which gives a brief insight of binary encoding capability. In general, DVB outperforms all state-of-the-art shallow and deep unsupervised methods with evident margins in most cases. Particularly, the minimum mAP gaps yield 1.5%, 0.7% and 1.6% on the three datasets respectively between DVB and other methods. It is clear that some existing unsupervised deep hashing models [22, 23] are no longer leading the retrieval performance compared with the shallow ones with deep features. Although benefited from the compact encoding neural networks, these deep methods still struggle in handling unsupervised hashing. This is probably because the batch-wise SGD procedure only manages to preserve the in-batch data similarities and therefore skews the statistics of the whole training set, which is empirically compensated in DVB by introducing the latent variables $\mathbf{z}$. UN-BDNN [8] obtains most acceptable performance among the existing deep methods, while it involves a more sophisticated optimization procedure than DVB. The Precision-Recall (PR) curves for image retrieval are illustrated in Figure 2 (a). To keep the paper reasonably concise, only 32-bit PR curves are reported here. The precision at top 5000 retrieval candidates (Precision@5000)

(a)



(b)

Figure 3: Examples of top-20 32-bit image retrieval results **(a)** and t-SNE [33] visualization **(b)** on CIFAR-10 [18].

Method	32-bit mAP	Training Time	Coding Time
DH [23]	0.176	152 minutes	20.7ms
DeepBit [22]	0.216	210 minutes	21.9ms
DVB	**0.365**	**127 minutes**	**29.4ms**

Table 2: Comparison of 32-bit training and encoding efficiency on CIFAR-10 [18] with some deep hashing methods.

Method	16 bits	32 bits	64 bits
Without $l2$ on $\mu_\theta(\cdot)$	0.269	0.325	0.342
Without $\mathcal{H}_2$	0.286	0.344	0.349
Without $\mathcal{H}_3$	0.317	0.350	0.363
DVB (full)	**0.347**	**0.365**	**0.381**

Table 3: Ablation study results (mAP) of DVB on CIFAR-10 [18] with some terms of the learning objective removed.

curves with all bits are plotted in Figure 2 (b) to have a more comprehensive view on retrieval performance.

The training and encoding time of DVB is demonstrated in Table 2, where DH [23] and DeepBit [22] are included for comparison. All experiments are conducted on an Nvidia TitanX GPU. DVB requires less training time than the two listed deep models since it takes less training epochs to reach the best performance. The test time of DVB tends to be slightly longer than DH [23] and DeepBit [22] but is still acceptable. This is because DVB involves a Monte Carlo multiple sampling procedure shown in Equation (12) to encode test data.

The retrieval performances of the three additional baselines for ablation study are shown in Table 3. We have experienced a significant mAP drop of 5% on average when omitting the $l2$ loss on $\mu_\theta(\cdot)$. It also can be observed that $\mathcal{H}_2$ and $\mathcal{H}_3$ do have a positive impact on the final result of DVB.

3.2 Qualitative results

Qualitative analysis is also provided to empirically demonstrate the binary encoding performance of DVB. Some intuitive retrieval results on 32-bit CIFAR-10 [18] are shown in Figure 3 (a), which suggests DVB is able to provide relative candidates in top of the retrieval sequences. The t-SNE [33] visualisation results on the test set of CIFAR-10 are illustrated in Figure 3 (b). It can be observed that the produced codes are not perfectly scattered on the two-dimensional panel as no class information is provided during parameter training. How-

ever, most classes are clearly segregated, which means the produced binary codes are still compact and semantically informative to some extent.

4 Conclusion

In this paper, a novel unsupervised deep hashing method DVB was proposed. The recent advances in deep variational Bayesian models have been leveraged to construct a generative model for binary coding. The latent variables in DVB approximate the pseudo data centres that each data point in the training set belongs to, by means of which DVB exploits the intrinsic structure of the dataset. By minimizing the gap between the constructed and reconstructed latent variables from data inputs and binary outputs respectively, the proposed model produces compact binary codes with no supervision. Experiments on three large-scale datasets suggested that DVB outperforms state-of-the-art unsupervised hashing methods with evident margins.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

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

[4] Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[5] Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *SIGIR*, 2017.

[6] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[7] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ACM-CIVR*, 2009.

[8] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *ECCV*, 2016.

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

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

[11] Yuchen Guo, Guiguang Ding, Li Liu, Jungong Han, and Ling Shao. Learning to hash with optimized anchor embedding for scalable retrieval. *IEEE Transactions on Image Processing*, 26(3):1344–1354, 2017.

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

[13] Xiaofei He and Partha Niyogi. Locality preserving projections. In *NIPS*, 2003.

[14] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *CVPR*, 2012.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[16] Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

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

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[19] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[20] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

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

[22] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, 2016.

[23] V. E. Liong, Jiwen Lu, Gang Wang, P. Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2015.

[24] Li Liu and Ling Shao. Sequential compact code learning for unsupervised image hashing. *IEEE transactions on neural networks and learning systems*, 27(12):2526–2536, 2016.

[25] Li Liu, Mengyang Yu, and Ling Shao. Unsupervised local feature hashing for image similarity search. *IEEE transactions on cybernetics*, 46(11):2548–2558, 2016.

[26] Li Liu, Zijia Lin, Ling Shao, Fumin Shen, Guiguang Ding, and Jungong Han. Sequential discrete hashing for scalable cross-modality similarity retrieval. *IEEE Transactions on Image Processing*, 26(1):107–118, 2017.

[27] Li Liu, Ling Shao, Fumin Shen, and Mengyang Yu. Discretely coding semantic rank orders for supervised image hashing. In *CVPR*, 2017.

[28] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *CVPR*, 2017.

[29] Li Liu, Mengyang Yu, and Ling Shao. Learning short binary codes for large-scale image retrieval. *IEEE Transactions on Image Processing*, 26(3):1289–1299, 2017.

[30] Li Liu, Mengyang Yu, and Ling Shao. Latent structure preserving hashing. *International Journal of Computer Vision*, 122(3):439–457, 2017.

[31] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, 2011.

[32] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, 2014.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[34] Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[35] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

[37] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7), 2009.

[38] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton Van Den Hengel, and Zhenmin Tang. Inductive hashing on manifolds. In *CVPR*, 2013.

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

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[41] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

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

[43] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, 2009.

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

[45] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[46] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016.

[47] Mengyang Yu, Li Liu, and Ling Shao. Structure-preserving binary representations for rgb-d action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1651–1664, 2016.

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

[49] Xiaofeng Zhu, Lei Zhang, and Zi Huang. A sparse embedding and least variance encoding approach to hashing. *IEEE Transactions on Image Processing*, 23(9):3737–3750, 2014.