5-2018

Binary Representation Learning for Large Scale Visual Data

Liu Liu
University of Tennessee, lliu25@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Liu Liu entitled "Binary Representation Learning for Large Scale Visual Data." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

Jens Gregor, Husheng Li, Russell L. Zaretzki

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Binary Representation Learning for Large Scale Visual Data

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Liu Liu

May 2018

This work is dedicated to my family, especially my mother, a rock upon which I stand, and an eternal beacon to which I march.

Acknowledgments

I would like to thank Dr. Hairong Qi, who is the most insightful and shrewd and patient genius you will ever work with. In addition, I would like to express my gratitude towards my labmates, who are the most diligent working bees and ingenious researchers; to my wonderful family, who supports me unconditionally.

Abstract

The exponentially growing modern media created large amount of multimodal or multi-domain visual data, which usually reside in high dimensional space. And it is crucial to provide not only effective but also efficient understanding of the data.

In this dissertation, we focus on learning binary representation of visual dataset, whose primary use has been hash code for retrieval purpose. Simultaneously it serves as multifunctional feature that can also be used for various computer vision tasks. Essentially, this is achieved by discriminative learning that preserves the supervision information in the binary representation.

By using deep networks such as convolutional neural networks (CNNs) as backbones, and effective binary embedding algorithm that is seamlessly integrated into the learning process, we achieve state-of-the art performance on several settings. First, we study the supervised binary representation learning problem by using label information directly instead of pairwise similarity or triplet loss. By considering images and associated textual information, we study the cross-modal representation learning. CNNs are used in both image and text embedding, and we are able to perform retrieval and prediction across these modalities. Furthermore, by utilizing unlabeled images from a different domain, we propose to use adversarial learning to connect these domains. Finally, we also consider progressive learning for more efficient learning and instance-level representation learning to provide finer granularity understanding. This dissertation demonstrates that binary representation is versatile and powerful under various circumstances with different tasks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern computer vision tasks such as **smart camera networks** (SCNs) [61] and **large-scale** visual data mining is becoming more and more ubiquitous and demanding. Computer vision tasks on huge collections of images and video are usually challenging due to its overwhelming size or high dimensionality, making recognition or similarity search inefficient and unaffordable. Many large-scale datasets such as ImageNet [38] have become available, enabling better studying more sophisticated algorithms. Meanwhile online sharing of user-generated content grows exponentially. For instance, Facebook has about 300 million photo uploads per day [22]. On the other hand, due to the resource-constrained nature of smart camera networks, often deployed in a distributed fashion with limited onboard processing, storage and transmission capacity, SCNs cannot handle large data transfer in typical applications like **distributed object/scene recognition** [Luo and Qi]. Moreover, modern media along with the large scale datasets generated by them brought forward following new challenges that traditional computer vision rarely dealt with:

1. rapidly growing social media offer massive volumes of multimedia content as well, e.g., photo posts with textual tags on Flickr, tweets with pictures, etc. It is desired to perform efficient content understanding and analytics across different media modalities.

2. Generating labels for large scale datasets is usually prohibitively expensive, and newly available datasets often do not come with label information, from which it is still desired to retrieve relevant data pertaining to labeled images. While this is achievable thanks to the transferability of deep structure [88], the non-negligible domain shift existing between different domains hinders more effective cross-domain image retrieval.

These high-demanding applications (e.g., SCNs and scalable data mining) are becoming more and more ubiquitous and renders renders it urgent to generate more efficient representations or descriptors of high fidelity for image datasets. Consequently learning high-quality binary representation is tempting due to its compactness and representation capacity.

The binary representation or image hashing, has been widely used in areas like massive data mining and large-scale machine learning, such as relevant information retrieval and similarity search [93, 27, 84, 55]. It maps high-dimensional and continuous valued data into compact binary codes, leading to considerable savings on both space (storage and

transmission) and time (computational complexity), thus becomes an ideal descriptor for representing large-scale datasets and solving resource-constrained problems in SCNs. Image hashing algorithms have been evolving from data-independent techniques [11] to data-driven methods, such as Spectral Hashing [84], Binary Reconstructive Embedding (BRE) [40], iterative quantization (ITQ) [20]. During the past decade, deep neural networks, such as autoencoder [33], restricted Boltzman machine (RBM) [74, 25] and convolutional neural network (CNN) [38, 82] have enabled the generation of highly semantic-preserving features. The recently developed VGG model [2] stacked over ten convolutional layers, generating high-level features and delivering outstanding classification performance. The deep residual network [29] (ResNet) pushed the limit of deep neural networks even further, resulting in networks of hundreds or even a thousand layers. Nonetheless, the real-valued features generated by the deep models are usually high-dimensional and are still too computationally heavy in applications like SCNs. Some recent studies [42, 90] attempted to leverage the deep models and were able to generate high-quality hash code. Majority of these approaches exploited the similarity information between samples for retrieval purpose. More specifically, this is realized by characterizing the similarity in a pre-defined neighborhood. Usually pairwise or triplet similarity are considered to capture such similarity among image pairs or triplets, respectively [56, 92, 51]. Although respecting the similarity semantics of the original dataset, the uniqueness of each individual is ignored, making it difficult to use the binary code to perform tasks like classification. Therefore it is very beneficial to generate effective binary representation that can be used for not only similarity search, indexing and retrieval, but also great for recognition and classification. Recently this gap was filled by several hashing algorithms that learn binary representation via classification [87, 75, 52]. Not only does the learned binary code retrieves images effectively, it provides comparable or even superior performance for classification as well. Meanwhile, due to the discrete nature of binary code, it is usually impractical to optimize discrete hashing function directly. Most hashing approaches attempt solving it by a continuous relaxation and quantization loss [75, 51]. However, such optimization is usually not statistically stable [92] and thus leads to suboptimal hash code.

Starting with the discussion of learning binary representation via classification tasks with cross entropy, this work focuses on the learning of discriminative binary representation for image datasets. Not only is the learned binary representation suitable for retrieval purpose, it also can be used as for classification tasks and annotation purposes, enabling learning multitasking representations. In this work, we first study the merits of learning binary representation for images using discriminative information instead of conventionally used similarity information. This is realized by utilizing label information directly with cross entropy as the loss function. Then we propose a novel architecture of binary embedding in deep neural network directly, enabling end-to-end learning of binary representation. Furthermore, we study the three scenarios where binary representation can be helpful: cross-view image hashing, cross-domain image understanding, and instance-level binary representation learning. We conduct extensive experiments to evaluate the proposed algorithms. Empirical evidences suggest that the proposed methods provide superior performance across various tasks.

Chapter 2

Learning Effective Binary Descriptors via Cross Entropy

A version of this chapter was originally published by Liu Liu and Hairong Qi:

Liu Liu, Hairong Qi, "Learning Effective Binary Descriptors via Cross Entropy", *IEEE Winter Conference on Applications of Computer Vision* (WACV) 2017.

2.1 Abstract

Binary descriptors not only are beneficial for similarity search, they are also capable of serving as discriminant features for classification purpose. In this paper we propose a new algorithm based on cross entropy to learn effective binary descriptors, dubbed CE-Bits, providing an alternative to L-2 and hinge loss learning. Because of the usage of cross entropy, a min-max binary NP-hard problem is raised to optimize the binary code during training. We provide a novel solution by breaking the binary code into independent blocks and optimize them individually. Although suboptimal, our method converges very fast and outperforms its L-2 and hinge loss counterparts. By conducting extensive experiments on several benchmark datasets, we show that CE-Bits efficiently generates effective binary descriptors for both classification and retrieval tasks and outperforms state-of-the-art supervised hashing algorithms.

2.2 Introduction

With the emergence of modern applications like **smart camera networks** (SCNs) [61] and **large-scale** data mining, computer vision tasks on huge collections of images and video are usually becoming more and more challenging due to its overwhelming size or high dimensionality, making recognition or similarity search inefficient and unaffordable. Many large-scale datasets such as ImageNet [38] have become available, enabling better studying more sophisticated algorithms. Meanwhile online sharing of user-generated content grows exponentially. For instance, Facebook has about 300 million photo uploads per day [22]. On the other hand, due to the resource-constrained nature of smart camera networks, often deployed in a distributed fashion with limited onboard processing, storage and transmission capacity, SCNs cannot handle large data transfer in typical applications like **distributed**

object/scene recognition [Luo and Qi]. These high-demanding applications (e.g., SCNs and scalable data mining) are becoming more and more ubiquitous and renders renders it urgent to have a significantly more efficient feature descriptor of high fidelity.

The binary code or hashing techniques, has been widely used in areas like massive data mining and large-scale machine learning [93, 27, 84, 55]. It maps high-dimensional and continuous valued data into compact binary codes, leading to considerable savings on both space (storage and transmission) and time (computational complexity), thus becomes an ideal descriptor for representing large-scale datasets and solving resource-constrained problems in SCNs. Image hashing algorithms have been evolving from data-independent techniques [11] to data-driven methods, such as Spectral Hashing [84], Binary Reconstructive Embedding (BRE) [40], iterative quantization (ITQ) [20]. During the past decade, deep neural networks, such as autoencoder [33], restricted Boltzman machine (RBM) [74, 25] and convolutional neural network (CNN) [38, 82] have enabled the generation of highly semantic-preserving features. The recently developed VGG model [2] stacked over ten convolutional layers, generating high-level features and delivering outstanding classification performance. The deep residual network [29] (ResNet) pushed the limit of deep neural networks even further, resulting in networks of hundreds or even a thousand layers. Nonetheless, the real-valued features generated by the deep models are usually high-dimensional and are still too computationally heavy in applications like SCNs. Some recent studies [42, 90] attempted to leverage the deep models and were able to generate high-quality hash code. However, majority of these approaches exploited the similarity information between samples for retrieval purpose, ignoring the uniqueness of each individual, making it difficult to use the binary code to perform tasks like classification. Therefore it is very beneficial to generate effective binary descriptors that can be used for not only similarity search, indexing and retrieval, but also great for recognition and classification.

Albeit the success of image hashing, learning effective binary descriptors is still an open-end topic. Although several efficient binary descriptors, e.g., BRIEF [7], BRISK [44], and FREAK [64], have been proposed to describe images without label information. However they tend to be unstable and inconsistent to the image invariant. Deep learning based binary descriptor [46] has been proposed to generate high-quality binary descriptor. But

it does not generalize the situation where existing continuous-valued features are already available. In this paper, we propose a new method to generate binary representations for images, which not only enjoys the high-quality features from deep neural networks, but also can be applied to resource-constrained environment like smart camera networks, where both computation and storage resources are restricted. Inspired by classic classification paradigm, we propose to use *cross entropy* as the loss function. Because of the use of cross entropy, we are faced with an NP-hard combinatorial optimization problem during training. We provide a suboptimal block-by-block greedy optimization algorithm for the binary codes. Empirical studies demonstrate that our training algorithm converges very fast. Moreover, experiment results also show that our method outperforms state-of-the-art supervised hashing algorithms, including its L-2 and hinge loss counterparts, on both classification and retrieval tasks.

2.3 Learning Binary Descriptors via Classification

Demonstrated by previous studies such as [75], binary descriptors learned via classification preserve the semantic similarity and discriminative information of the original data and serve multiple purposes. Not only can they be used for classification, they are also good hash code for retrieval task. Similar to the setup in [75], the binary descriptors are learned via classification. Specifically, given a dataset of N samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$, we aim to learn the binary representation for the dataset, denoted by $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N} \in \{-1, +1\}^{L\times N}$, which is obtained by taking the sign of a learned embedding function $F : \mathbb{R}^{d\times N} \mapsto \mathbb{R}^{L\times N}$, $L << d$:

$$\mathbf{B} = \text{sgn}(F(\mathbf{X})) \tag{2.1}$$

Meanwhile a linear classifier is used to take the advantage of the label information associated with $\mathbf{X}$, namely $\mathbf{Y}$, where $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}, y_i = \{1, \ldots, C\}, i = 1, \ldots, N$, and C is the number of categories. The weight of the classifier is denoted as $\mathbf{W} \in \mathbb{R}^{L\times C}$. The binary codes are

obtained by using the following optimization problem similar to [75]:

$$\begin{aligned} &\min_{\mathbf{W},F} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathbf{W}^T\mathbf{b}_i, y_i) + \Omega(\mathbf{W}) \\ &\mathbf{b}_i = \text{sgn}(F(\mathbf{x}_i)) \in \{-1,+1\}^L, i = 1,\ldots,N \end{aligned} \tag{2.2}$$

where $\mathcal{L}$ is some loss function measuring the error between the prediction and ground truth $\mathbf{Y}$ during training, and Ω is the regularizor for the classifier.

Common choices for loss function are L-2 loss (dictionary learning [78] and supervised modeling [25]), hinge loss (SVM), and cross entropy (neural networks). With L-2 loss and hinge loss explored in [75], in this paper we choose to use cross entropy as the loss function as it raises a new optimization problem, leading to a more efficient solution for generating the binary descriptors as demonstrated by extensive experiments in Section 4.5. Consequently the classifier we use here is *softmax*. Softmax is a generalization of binary logistic regression classifier, which tends to give a more intuitive result in a probabilistic sense. The weight of the classifier $\mathbf{W}$ can be expanded as $\{\mathbf{w}_1,\ldots,\mathbf{w}_C\}$ where $\mathbf{w}_k \in \mathbb{R}^{L\times 1}$ is for category k. Unnormalized probability of prediction for category k can then be expressed as $e^{\mathbf{w}_k^T\mathbf{b}_i}$. And the normalized probability of prediction is:

$$P(y_i = k|\mathbf{b}_i;\mathbf{w}_k) = \frac{e^{\mathbf{w}_k^T\mathbf{b}_i}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^T\mathbf{b}_i}} \tag{2.3}$$

Cross entropy aims to minimize the difference of probability distribution between the predicted labels and ground truth labels. For each sample we have:

$$\mathcal{L}_i = -\sum_{k=1}^{C} t_k(y_i)\log P(y_i = k|\mathbf{b}_i;\mathbf{w}_k), \tag{2.4}$$

where $t_k(y_i)$ is the distribution of ground truth labels y_i; $P(y_i = k|\mathbf{b}_i;\mathbf{w}_k)$ is the distribution of predicted labels. This is equivalent to Kullback-Leibler divergence since the entropy of the ground truth is a constant and does not contribute, and both of which attempt to minimize the distance between two distributions. Note that the distribution of ground truth label t_k can be simplified as identity function since each training sample only belongs to

one category k, i.e., $\sum_{k=1}^{C} t_k(y_i) = \mathbb{1}(y_i = k)$, furthermore simplified as $\mathbb{1}(y_i)$. Substitute Eq. 2.4 into Eq. 2.2 and use Frobenius norm regularization as Ω, we formulate the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{b}_i, \mathbf{W}, F} -\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i) \left(\mathbf{w}_k^T \mathbf{b}_i - \log \sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{b}_i} \right) + \lambda \|\mathbf{W}\|_F^2 \\ &\mathbf{b}_i = \operatorname{sgn}(F(\mathbf{x}_i)), i = 1, \ldots, N \end{aligned} \tag{2.5}$$

It is difficult to optimize Eq. 2.5 directly due to the discontinuity introduced by $\operatorname{sgn}(\cdot)$. Following the relaxation in [75], we add a penalty that accounts for the deviation between the continuous embedding function F and the binary code $\mathbf{B}$. Eq.2.5 is reformulated as

$$\begin{aligned} &\min_{\mathbf{b}_i, \mathbf{W}, F} \left\{ -\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i) \left(\mathbf{w}_k^T \mathbf{b}_i - \log \sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{b}_i} \right) + +\lambda \|\mathbf{W}\|_F^2 + \gamma \sum_{i=1}^{N} \|\mathbf{b}_i - F(\mathbf{x}_i)\|_2^2 + \rho \|F\|_2^2 \right\} \\ &\text{s.t.} \quad \mathbf{b}_i \in \{-1, +1\}^L, i = 1, \ldots, N. \end{aligned} \tag{2.6}$$

where γ is the coefficient for the regularization of the embedding function F.

Since there are multiple sets of parameters to learn, Eq.2.6 can be solved iteratively set by set. As mentioned before, because of the usage of the cross entropy as the loss function, the problem becomes NP-hard. We share the same paradigm as outlined in [75] where the F step is the same, but the W step and B step need to be redesigned.

2.3.1 F step: embedding function optimization

Following [56, 75], we use a very popular and powerful nonlinear embedding mapping function of the form

$$F(\mathbf{x}) = \mathbf{M}^T \phi(\mathbf{x}) \tag{2.7}$$

where $\mathbf{M} \in \mathbb{R}^{m \times L}$ is a linear mapping; $\phi(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1), K(\mathbf{x}, \mathbf{x}_2), \cdots, K(\mathbf{x}, \mathbf{x}_m)]^T$. $K(\mathbf{x}, \mathbf{x}_i)$, $i = 1, \cdots, m$ is a kernel function and points $\{\mathbf{x}_i\}_{i=1}^{m}$ are anchors uniformly sampled from training set. A popular choice for kernel function is the RBF kernel function $K(\mathbf{x}, \mathbf{x}_i) = e^{\|\mathbf{x} - \mathbf{x}_1\|^2 / \sigma}$, where σ is the kernel width controlling the shape of the kernel function.

In order to optimize the embedding function parameterized by $\mathbf{M}$, we fix the binary code B and classifier W and we have

$$\begin{aligned} &\min_{\mathbf{M}} \sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{M}^T \phi(\mathbf{x}_i)\|_2^2 + \rho \|\mathbf{M}\|_2^2 \\ &\text{s.t.} \quad \mathbf{b}_i = \{-1, +1\}^L. \end{aligned} \tag{2.8}$$

Eq. 2.8 can be rewritten in matrix form

$$\begin{aligned} &\min_{\mathbf{M}} \|\mathbf{B} - \mathbf{M}^T \phi(\mathbf{X})\|_2^2 + \rho \|\mathbf{M}\|_2^2 \\ &\text{s.t.} \quad \mathbf{B} = \{-1, +1\}^{L \times N}. \end{aligned} \tag{2.9}$$

Eq. 2.9 can be solved by the regularized least square

$$\mathbf{M} = (\phi(\mathbf{X})\phi(\mathbf{X})^T + \rho \mathbf{I})^{-1} \phi(\mathbf{X})\mathbf{B} \tag{2.10}$$

2.3.2 W step: classifier optimization

By fixing binary code $\mathbf{B}$ and the embedding function F, classifier can be optimized by solving the following problem:

$$\min_{\mathbf{W}} -\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i) \log \frac{e^{\mathbf{w}_k^T \mathbf{b}_i}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^T b_i}} + \lambda \|\mathbf{W}\|_F^2, \tag{2.11}$$

which can be solved through gradient based approaches. The gradient of the objective function in Eq. 2.11 with respect to each $\mathbf{w}_k$ associated with category k is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k} = -\frac{1}{N} \sum_{i=1}^{N} \left[\mathbf{b}_i \left(\mathbb{1}(y_i) - \frac{e^{\mathbf{w}_k^T \mathbf{b}_i}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{b}_i}} \right) \right] + 2\lambda \mathbf{w}_k \tag{2.12}$$

In order to accelerate gradient descent in the relevant direction and dampen the oscillation phenomenon during learning process, momentum [67] is used. Then the updating rule for

$\mathbf{w}_k$ is

$$\begin{aligned} v^{(t)} &= \theta v^{(t-1)} + \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}_k^{(t-1)}} \\ \mathbf{w}_k^{(t+1)} &= \mathbf{w}_k^{(t)} - v^{(t)}, \quad \forall k = 1, \ldots, C \end{aligned} \tag{2.13}$$

where θ is the momentum term, usually set to 0.9 or smaller, α is the learning rate.

2.3.3 B step: binary code optimization

Similarly $\mathbf{B}$ is optimized by fixing $\mathbf{W}$ and F (defined in Eq. 5.10), and we have the following optimization problem,

$$\begin{aligned} \min_{\mathbf{b}_i} & \left\{ -\left(\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i) \mathbf{w}_k^T \mathbf{b}_i + 2\gamma \sum_{i=1}^{N} F(\mathbf{x}_i)^T \mathbf{b}_i \right) + \frac{1}{N} \sum_{i=1}^{N} \log \sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{b}_i} \right\} \\ \text{s.t.} \quad & \mathbf{b}_i \in \{-1, 1\}^L, i = 1, \ldots, N. \end{aligned} \tag{2.14}$$

where $\log \sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{b}_i}$ in problem (2.14) is a *Log-Sum-Exp* (LSE) function, which is a smooth approximation to the maximum function, owing to the following tight bounds,

$$\max\{x_1, \ldots, x_n\} \leq LSE(x_1, \ldots, x_n) \leq \max\{x_1, \ldots, x_n\} + \log(n) \tag{2.15}$$

When $\{x_i\}_1^n$ are large enough, LSE can be approximated directly by the maximum function. Here even if $x_i = \mathbf{w}_k^T \mathbf{b}_i$ is not large enough, we can still accomplish the approximation since $LSE(\mathbf{w}_k^T \mathbf{b}_i) = -m + \log \sum_{i=1}^{N} \exp(\mathbf{w}_k^T \mathbf{b}_i + m)$, where m is a large enough number. In fact, many use such trick to prevent numerical overflow of calculating $LSE(x_n)$ in practice and ususally $m = \max_n(x_n)$ in such case.

As a result, Eq. 2.14 can be approximated as

$$\begin{aligned} \min_{\mathbf{b}_i} & \left\{ -\left(\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i) \mathbf{w}_k^T \mathbf{b}_i + 2\gamma \sum_{i=1}^{N} F(\mathbf{x}_i)^T \mathbf{b}_i \right) + \frac{1}{N} \sum_{i=1}^{N} \max_j \{\mathbf{w}_j^T \mathbf{b}_i\} \right\} \\ \text{s.t.} \quad & \mathbf{b}_i \in \{-1, +1\}^L, i = 1, \ldots, N. \end{aligned} \tag{2.16}$$

Clearly Eq. 2.16 is an NP-hard combinatorial problem. We choose to solve $\mathbf{b}_i$ by exhaustively searching in the solution space. Since $\mathbf{B} \in \{-1, 1\}^{L \times N}$, the worst computational complexity

is $O(2^{NL})$. Usually it is impossible to solve the problem if we directly search the whole solution space. The following two observations inspire our suboptimal solution. *The input images can be considered as independent samples (i.i.d for each category); meanwhile each bit of the binary code can be treated as a random variable following the same Bernoulli distribution, and we assume the bits are statistically independent.* As a matter of fact, we can exchange the order of the sum and minimization of Eq. 2.16

$$\begin{aligned}&\sum_{i=1}^{N}\left(\min_{\mathbf{b}_i}\left\{-\left(\frac{1}{N}\mathbb{1}(y_i)\mathbf{w}_k^T\mathbf{b}_i+2\gamma F(\mathbf{x}_i)^T\mathbf{b}_i\right)+\frac{1}{N}\max_j\{\mathbf{w}_j^T\mathbf{b}_i\}\right\}\right)\\&\text{s.t.}\quad \mathbf{b}_i\in\{-1,1\}^L, i=1,\ldots,N.\end{aligned}\tag{2.17}$$

So the binary hash code for each sample can be optimized independently. Denoting the optimization problem 2.17 for ith sample's binary code $\mathbf{b}_i$ as $J(\mathbf{b}_i)$, by "divide and conquer", we can solve $J(\mathbf{b}_i)$ in a greedy way. To see this, we decompose the problem into two sub-problems by splitting the binary code $\mathbf{b}_i$ into two halves $\mathbf{b}_{i,1}$ and $\mathbf{b}_{i,2}$:

$$\begin{aligned}&\min_{\mathbf{b}_i}J(\mathbf{b}_i)\\&=\min_{\mathbf{b}_{i,1},\mathbf{b}_{i,2}}\left\{-\frac{1}{N}\left(\mathbb{1}(y_i)\mathbf{w}_{k,1}^T\mathbf{b}_{i,1}+\mathbb{1}(y_i)\mathbf{w}_{k,2}^T\mathbf{b}_{i,2}\right)-2\gamma\left(F_1(\mathbf{x}_i)^T\mathbf{b}_{i,1}+F_2(\mathbf{x}_i)^T\mathbf{b}_{i,2}\right)\right.\\&\left.+\frac{1}{N}\max_j\{\mathbf{w}_{j,1}^T\mathbf{b}_{i,1}+\mathbf{w}_{j,2}^T\mathbf{b}_{i,2}\}\right\}\\\approx&\min_{\mathbf{b}_{i,1},\mathbf{b}_{i,2}}\left\{-\frac{1}{N}\left(\mathbb{1}(y_i)\mathbf{w}_{k,1}^T\mathbf{b}_{i,1}+\mathbb{1}(y_i)\mathbf{w}_{k,2}^T\mathbf{b}_{i,2}\right)-2\gamma\left(F_1(\mathbf{x}_i)^T\mathbf{b}_{i,1}+F_2(\mathbf{x}_i)^T\mathbf{b}_{i,2}\right)\right.\\&\left.+\frac{1}{N}\max_j\{\mathbf{w}_{j,1}^T\mathbf{b}_{i,1}\}+\frac{1}{N}\max\{\mathbf{w}_{j,2}^T\mathbf{b}_{i,2}\}\right\}\\=&\min_{\mathbf{b}_{i,1}}J(\mathbf{b}_{i,1})+\min_{\mathbf{b}_{i,2}}J(\mathbf{b}_{i,2})\end{aligned}\tag{2.18}$$

Now we have two separate optimization problems with the identical form. And this splitting process continues until solving the sub-problem by exhaustive search is affordable. The process is shown in Fig. 5.2. More specifically, the pth block of the ith sample can be optimized by $J(\mathbf{b}_{(i,p)})$, which can be solved in $O(\log C+2^{L_0})$ due to finding maximum of $\mathbf{W}_{(p,:)}^T\mathbf{b}_{(i,p)}$, where L_0 is the size of a block. We can choose an affordable L_0 as the width of binary code in which the sub-problem we want to solve, yielding a runtime

complexity $O(2^{L_0} N \cdot \lceil \frac{L}{L_0} \rceil \log C)$, equivalent to $O(NL \log C)$ since L_0 is a constant. Despite the suboptimality of the solution, it provides an efficient, yet still effective approach to tackle the NP-hard problem. Empirically we set L_0 to 4, providing both high performance and efficient training.

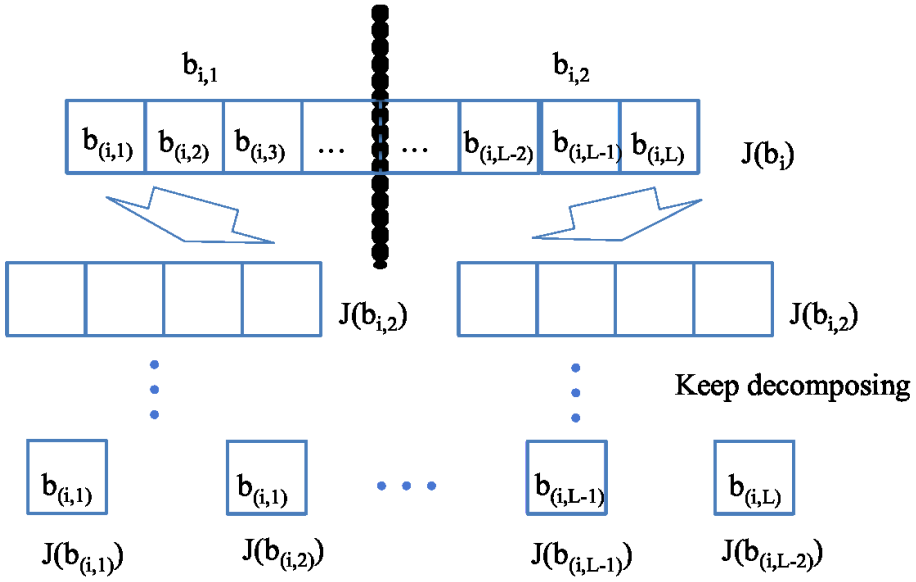


Figure 2.1: The procedure of decomposition of the problem using dynamic programming. The L-bit long optimization problem $J(\mathbf{b}_i)$ can be decomposed into bit-wise optimization problem $J(\mathbf{b}_{(i,j)})$, sharing exactly the same formulation of the original problem

The proposed CE-Bits is summarized in Algorithm 1.

2.4 Experiments

We evaluate the proposed CE-Bits on three challenging datasets: CIFAR-10, the Berkeley multiview wireless (BMW) and the Oxford 17 category flowers [62]. CIFAR-10 serves as a large dataset benchmark while BMW and Oxford 17 category flowers datasets are used in the smart camera network scenario. On each dataset we compare our algorithm with state-of-the-art supervised hashing algorithms (KSH [56], FasthHash [45], SDH [75], CCA-ITQ [20]) and provide extensive experiments on image classification task as well as image retrieval task. For the classification task, we split each dataset into a training set, and a testing set. In order to select an appropriate set of parameters for CE-Bits, we further randomly select a small set as the validation set. The performance of the classification task is evaluated by

Algorithm 1 Cross Entropy Hashing Classification

Input: Training set $\{\mathbf{X}, \mathbf{Y}\}$; code width L; number of iterations R; learning rate α; parameter λ, ν.

Output: Classifier weight $\mathbf{W}$; binary hash code matrix $\mathbf{B}$; hash function $F(\cdot)$; prediction of classification

Step 1: Initialization

initialize $\mathbf{W}$, $\mathbf{B}$ randomly; randomly pick anchors from training set; store solution space $\mathbf{b}_0 = [-1, +1]^T$ for exhaustive search; initialize $F(\mathbf{x})$

while $i < R$ **do**

$i = i + 1$

(B Step)

for $j = 1, \ldots, N$ **do**

for $k = 1, \ldots, L$ **do**

$\mathbf{m}_1 \leftarrow \max_c\{\mathbf{w}_c^T \mathbf{b}_{base}/N\}$

$\mathbf{m}_2 \leftarrow (\mathbf{W}_{(k,\mathbf{Y}(j))}^T/N + \gamma * F(\mathbf{x})_{j,k})\mathbf{b}_0$

$\mathbf{B}_{j,k} \leftarrow \arg\min_{-1,+1}\{\mathbf{m}_1 + \mathbf{m}_2\}$

end for

end for

(G Step)

$v_t = \theta v_{t-1} + \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}_k}$

$\mathbf{w}_k = \mathbf{w}_k - v_t, \quad \forall k = 1, \ldots, C$

(F Step)

$\mathbf{P} \leftarrow (\phi(\mathbf{X})\phi(\mathbf{X})^T)^{-1}\phi(\mathbf{X})\mathbf{B}^T$

end while

the accuracy on the testing set. Based on the model trained by the classification task, the retrieval task treats testing set as the query set and training set as the retrieval database. The performance of the retrieval task is evaluated by the mean average precision (MAP). In the experiments we use the recommended parameters for the compared algorithms.

2.4.1 Datasets

The CIFAR-10 dataset consists of 60,000 color images in 10 classes. Each class has 6,000 images in size 32×32. We randomly select 50,000 samples as the training set and the rest 10,000 as the testing set. And a validation set is split by randomly selecting 5,000 samples from the training set. Due to recent huge success achieved by deep neural network on large datasets like CIFAR-10, we represent the dataset using the output from the finally fully-connected layer of a 50-layer deep residual network, which is specifically trained on CIFAR-10. Each sample is a 64-dimension floating number vector.

The BMW dataset contains 20 landmark buildings on the campus of the University of California, Berkeley. 16 different vantage points are selected for each building to capture the 3-D appearance of the building. At each vantage point 5 short-baseline images are taken by 5 cameras simultaneously, leading to 80 images per category. And each image is a 640×480 RGB color image. For the experiments, we use images captured by camera #2 (320 images) as the training dataset (a validation set of 4 randomly selected samples per class is used for parameter selection), and rest images are testing dataset. Each image is described by a 500-dimension bag-of-words SURF [5] features.

The Oxford 17 category flower dataset [62] contains images of 17 categories of flowers. There are totally 1,380 images with each class consists of 80 images. A training set with 40 images per class, a validation set with 20 images per class and a testing set with 20 images per class are split. Instead of using the raw pixels ($227 \times 227 \times 3$), we use the pre-trained VGG model [2] (trained on the ImageNet dataset [12]) and obtain a 4096-dimension floating number vector for each image.

The data samples of all three datasets are preprocessed by normalizing to unit length.

2.4.2 Classification Task

Here we provide the comparison of testing accuracy on three datasets. In order to make a fair comparison, liblinear [15] is used as the classifier to assess the quality of the binary codes on classification task. Results with 64 bits are reported across all the methods. We also report the training time of the algorithms to compare the efficiency. Table 3.3, 2.2, 2.3 shows the results on CIFAR-10, Oxford 17 category flower [62] and BMW respectively. Note that due to fact that KSH requires huge memory and long time to train, a 5,000-sample training subset is randomly selected from the original 50,000-sample training set. To provide fair comparison, we also provide the result of CE-Bits that is trained on the same subset.

Table 2.1: The testing accuracy of different methods on CIFAR-10 dataset (ResNet features), all binary codes are 64 bits.

Methods	Testing Accuracy	Training Time (sec)
KSH (5,000 tr) [56]	91.5%	1720
FastHash [45]	92.3%	609
SDH [75]	92.0%	33.4
CCA-ITQ [20]	91.8%	**3.2**
ResNet Feature [29]	**92.4**%	-
CE-Bits (5,000 tr)	92.1%	3.1
CE-Bits	**92.4**%	22.1

Table 2.2: The testing accuracy of different methods on Oxford 17 category flower dataset [62] (VGG features), all binary codes are 64 bits.

Methods	Testing Accuracy	Training Time (sec)
KSH [56]	87.4%	83.1
FastHash [45]	88.5%	38.0
SDH [75]	87.9%	**0.71**
CCA-ITQ [20]	88.5%	7.67
VGG Feature [2]	**88.8**%	-
CE-Bits	**88.6**%	1.12

For all three challenging datasets, CE-Bits achieves the best accuracy among all the state-of-the-art supervised hashing algorithms. We can conclude that CE-Bits preserves

Table 2.3: The testing accuracy of different methods on BMW dataset (SURF features), all binary codes are 64 bits.

Methods	Testing Accuracy	Training Time (sec)
KSH [56]	93.8%	18.4
FastHash [45]	91.1%	14.8
SDH [75]	95.9%	**0.15**
CCA-ITQ [20]	92.9%	1.17
SURF [5]	94.7%	-
CE-Bits	**97.2**%	0.31

the discriminant information of the original floating-number data. For CIFAR-10 dataset, CE-Bits achieves the same best accuracy as the floating-number residual network features with a very low training time. Not only does this demonstrates that CE-Bits preserves the semantics of the ResNet features, it also implies the significant level of redundancy in the original floating-number features. Despite the fact that CCA-ITQ uses the least time to train the model, the testing accuracy is lower than CE-Bits. In addition, CCA-ITQ is sensitive to the dimension of the input, i.e., it achieves the lowest training time solely because the residual network feature of CIFAR-10 is only 64-dimension.

For Oxford 17 category flower dataset [62], CE-Bits delivers the best binary code classification accuracy with a very low training time with much less data, and the result is only slightly lower (0.2% lower) than the VGG [2] feature, which is 4096-dimension floating number. Similarly for BMW dataset, CE-Bits uses very low training time and achieves the best binary code classification accuracy. Note that the accuracy achieved by CE-Bits is even better than the original floating-number feature, also reported in [75], indicating that CE-Bits can extract more discriminant information. This is because the embedding function F maps the original feature to a nonlinear yet simpler feature space, enabling a high-quality binary descriptor.

2.4.3 Retrieval Task

We use CIFAR-10 as the benchmark dataset to evaluate the retrieval performance as it is usually much more challenging to do retrieval on large dataset like CIFAR-10. More

specifically, precision and mean average precision (MAP) within Hamming radius of 2 are used to evaluate the retrieval performance. Fig. 2.2 shows the comparison on precision of different methods. Based on the precision comparison, CE-Bits outperforms other state-of-the-art methods slightly across all code widths. Note that code width of CCA-ITQ is bound by the deep residual network feature of CIFAR-10, which is 64-dimension.

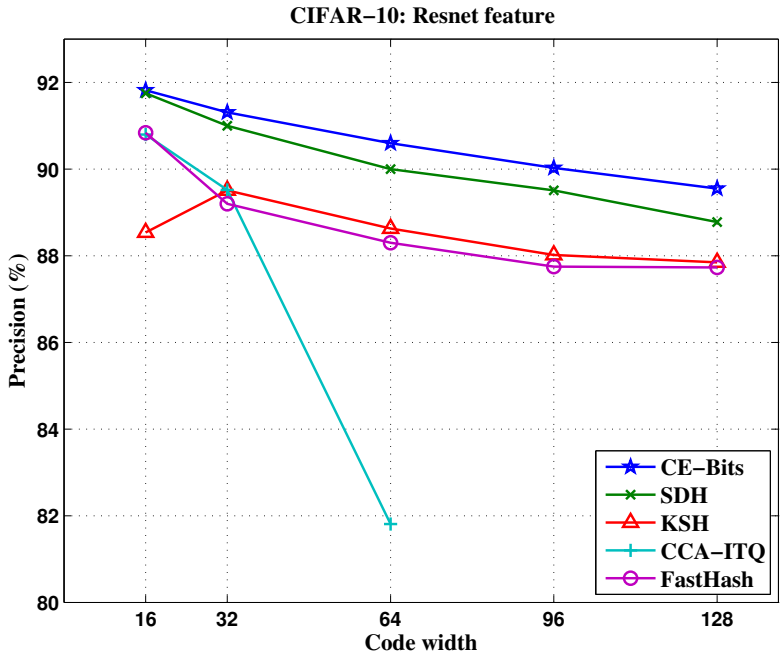


Figure 2.2: Comparison of precision achieved by different methods within Hamming radius of 2.

The comparison on MAP is demonstrated in Fig. 2.3. CE-Bits performs consistently well across all code widths in terms of MAP and it provides comparable results comparing to other state-of-the-art methods.

Note that we use different approaches (SURF, VGG, and ResNet) to generate features for the purpose of showing that CE-Bits can learn high-quality binary code consistently. If we use ResNet for all three datasets, CE-bits outperforms state-of-the-art algorithms as well. For Oxford 17 category flowers dataset, CE-Bits achieves classification accuracy of 94.76% and MAP of 95.47%, outperforming SDH (accuracy 93.26%, and MAP 94.59%). For BMW dataset, CE-Bits achieves accuracy of 98.02% and MAP of 99.26%, improving SDH (accuracy 97.96%, and MAP 98.57%).

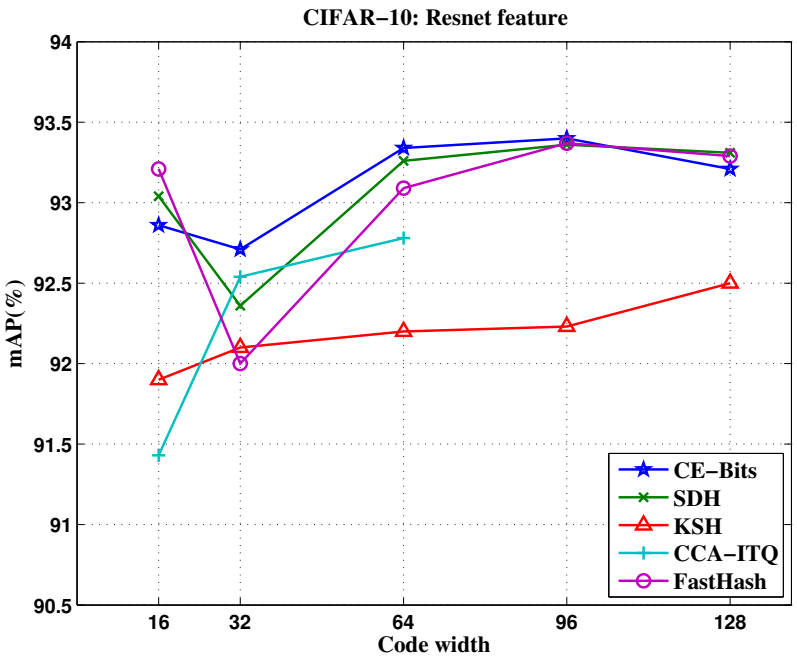


Figure 2.3: Comparison of MAP achieved by different methods within Hamming radius of 2.

2.4.4 Discussion

The behavior of the proposed CE-Bits is analyzed.

Suboptimality

We use CIFAR-10 as our benchmark dataset. Since the binary code $\mathbf{B}$ is optimized by breaking down into smaller blocks and optimizing them independently, obviously the solution is suboptimal, and the block size L_0 has a great impact on the effectiveness and the efficiency of the algorithm. On one hand, the greater L_0 is the closer the solution approaches to the optimal; on the other hand, larger L_0 can lead to substantially longer training time because the complexity of exhaustive search is proportional to 2^{L_0}. Tab. 2.4 summarizes how L_0 effects the algorithm. Surprisingly with smaller L_0, e.g., 1-bit and 2-bit, the training time is longer than L_0 =4-bit. This is because the exhaustive search has to loop through $\frac{L}{L_0}$ blocks, and the smaller L_0 is, the more blocks the algorithm has to optimize.

Table 2.4: Evaluation of suboptimality on different block size L_0. The code width is 64-bit.

L_0	1-bit	2-bit	4-bit	8-bit	16-bit
Testing accuracy (%)	91.5	92.0	92.4	92.3	92.4
Training Time (sec)	81	50.2	22.1	30.1	1105

Empirical Convergence

In the training stage, the derivation of embedding function has a closed form; and solving binary code is done by exhaustive search. The only factor that would affect the convergence is learning the classifier weight $\mathbf{W}$. However with carefully chosen learning rate α, CE-Bits converges fast and usually it only needs fewer than 5 iterations to converge. Fig. 2.4 shows that the convergence of CE-Bits on CIFAR-10 dataset is very fast. The learning rate for CIFAR-10 is $\alpha = 5e-3$, and for BMW as well as Oxford 17 category flower is $\alpha = 5e-2$.

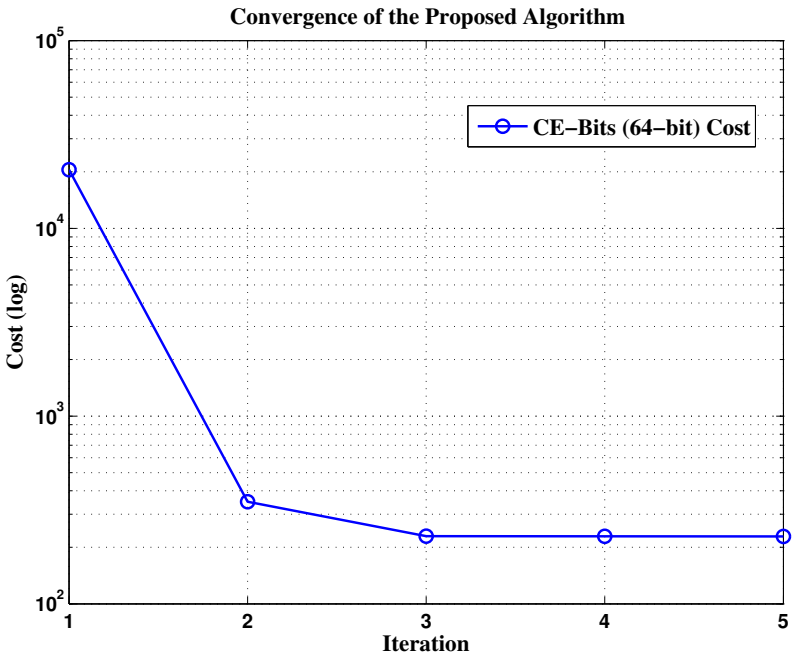


Figure 2.4: The convergence of CE-Bits on CIFAR-10 during training with learning rate $\alpha = 5e-3$. The code width is 64-bit

Anchors

By using randomly selected anchors, the dataset is projected to a nonlinear space by the embedding function. Although the impact of the number of anchors has been discussed in previous studies [56, 75], we demonstrate that this impact is actually data-related. Fig. 2.5 displays the impact of the number of anchors on BMW dataset and CIFAR-10. For BMW dataset, increasing the number of anchors significantly improves the performance of the algorithm; while the performance on CIFAR-10 is more consistent over different number of anchors since the residual network feature is more informative and robust comparing to the traditional hand-crafted features like GIST or SURF feature.

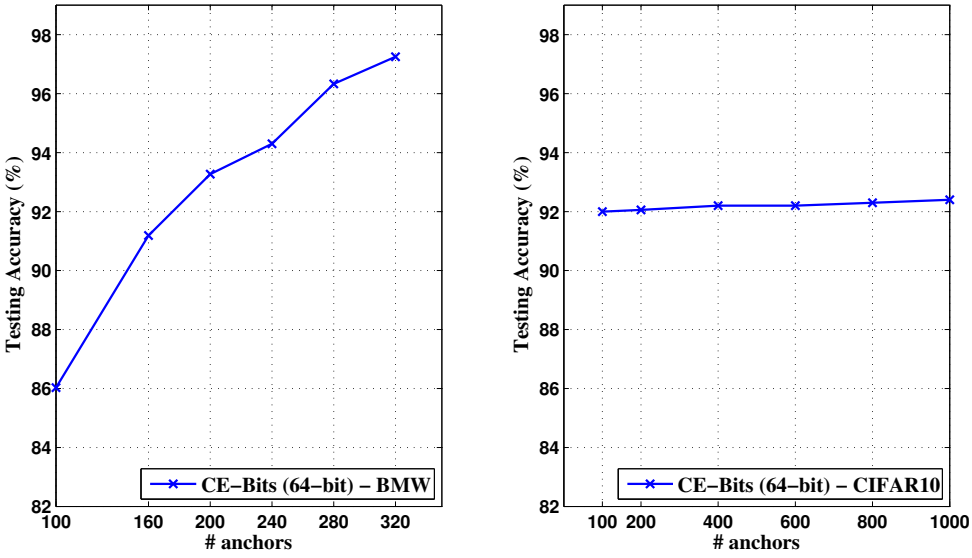


Figure 2.5: The testing accuracy of CE-Bits (64-bit) on BMW and CIFAR-10 regarding various number of anchors

Benefits from Binary Codes

Clearly using binary codes to represent images saves tremendous storing space and data transmission. For instance, storing and transmitting the data for BMW dataset on smart camera sensors with the SURF feature requires about 2K Bytes for each image; while it only needs 64 bits using binary codes, only 0.4% of original space to store or transmit the data. The storage required by the binary descriptors for Oxford 17 category flower dataset [62]

is even smaller, only 0.05% of the original VGG [2] feature. Meanwhile, because of the simplicity of binary codes, the computational cost is reduced drastically too. Take CIFIAR-10 dataset for example, using linear-SVM to train and test on the original residual network dataset takes 6.617 sec while it only takes 0.0069 sec on binary codes, yielding 1,000x faster calculation.

2.5 Conclusion

In this study, we proposed a new algorithm, dubbed CE-Bits, for generating effective binary descriptor, especially for computer vision task in extreme context. Based on classic formulation of classification, our algorithm is straightforward conceptually. Cross entropy is chosen as the criterion to formulate the optimization problem. We were able to show the compact binary descriptors can be generated effectively and efficiently by extensive experiments on three challenging experiments, CIFAR-10, Oxford 17 category flower, and Berkeley multiview wireless dataset. CE-Bits outperformed other state-of-the-art algorithms consistently for handcrafted features (SURF) and deep features (VGG and ResNet), while it required less training time especially on larger datasets.

Chapter 3

End-to-end Binary Representation Learning via Direct Binary Embedding

A version of this chapter was originally published by Liu Liu, Alireza Rahimpour, Ali Taalimi, Hairong Qi:

Liu Liu, Alireza Rahimpour, Ali Taalimi, Hairong Qi, "End-to-end Binary Representation Learning via Direct Binary Embedding", *IEEE International Conference on Image Processing* (ICIP) 2017

3.1 Abstract

Learning binary representation is essential to large-scale computer vision tasks. Most existing algorithms require a separate quantization constraint to learn effective hashing functions. In this work, we present Direct Binary Embedding (DBE), a simple yet very effective algorithm to learn binary representation in an end-to-end fashion. By appending an ingeniously designed DBE layer to the deep convolutional neural network (DCNN), DBE learns binary code directly from the continuous DBE layer activation without quantization error. By employing the deep residual network (ResNet) as DCNN component, DBE captures rich semantics from images. Furthermore, in the effort of handling multilabel images, we design a joint cross entropy loss that includes both softmax cross entropy and weighted binary cross entropy in consideration of the correlation and independence of labels, respectively. Extensive experiments demonstrate the significant superiority of DBE over state-of-the-art methods on tasks of natural object recognition, image retrieval and image annotation.

3.2 Introduction

Representation learning is key to computer vision tasks. Recently with the explosion of data availability, it is crucial for the representation to be computationally efficient as well [75, 52, 68]. Consequently learning high-quality binary representation is tempting due to its compactness and representation capacity.

Binary representation traditionally has been learned for image retrieval and similarity search purposes (image hashing). From the early works using hand-crafted visual features [20, 84, 56, 45] to recent end-to-end approaches [92, 51, 87] that take advantages of deep convolutional neural networks (DCNN), the core of image hashing is learning binary

code for images by characterizing the similarity in a pre-defined neighborhood. Usually pairwise or triplet similarity are considered to capture such similarity among image pairs or triplets, respectively [56, 92, 51]. Albeit the high-quality of binary code, most image hashing algorithms do not consider learning discriminative binary representation. Recently this gap was filled by several hashing algorithms that learn binary representation via classification [87, 75, 52]. Not only does the learned binary code retrieves images effectively, it provides comparable or even superior performance for classification as well. Meanwhile, due to the discrete nature of binary code, it is usually impractical to optimize discrete hashing function directly. Most hashing approaches attempt solving it by a continuous relaxation and quantization loss [75, 51]. However, such optimization is usually not statistically stable [92] and thus leads to suboptimal hash code.

In this work, we propose to learn high-quality binary representation *directly* from deep convolutional neural networks (DCNNs). By appending a binary embedding layer directly into the state-of-the-art DCNN, deep residual network, we train the whole network as a hashing function via classification task in attempt to learning representation that approximates binary code without the need of using quantization error. Thus we name our approach Direct Binary Embedding (DBE). Furthermore, in order to learn high-quality binary representation for multilabel images, we propose a joint cross entropy that incorporates softmax cross entropy and weighted binary sigmoid cross entropy in consideration of the correlation and independence of labels, respectively. Extensive experiments on two large-scale datasets (CIFAR-10 and Microsoft COCO) show that the proposed DBE outperforms state-of-the-art hashing algorithms on object classification and retrieval tasks. Additionally, DBE provides a comparable performance on multilabel image annotation tasks where usually continuous representation is used.

3.3 Direct Binary Embedding

3.3.1 Direct Binary Embedding (DBE) Layer

We start the discussion of DBE layer by revisiting learning binary representation using classification. Let $\mathbf{I} = \{I_i\}_{i=1}^{N}$ be the image set with n samples, associated with label set

$\mathbf{Y} = \{y_i\}_{i=1}^N$. We aim to learn binary representation $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N \in \{0, +1\}^{N \times L}$ of $\mathbf{I}$ via the Direct Binary Embedding layer that is appended to DCNN. Following the paradigm of classification problem formulation in DCNN, we use a linear classifier $\mathbf{W}$ to classify the binary representation:

$$\begin{aligned} &\min_{W,F} \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{L}(\mathbf{W}^\top \mathbf{b}_i, y_i) + \lambda \|\mathbf{b}_i - F(I_i; \Omega)\|_2^2 \right) \\ &\text{s.t. } \mathbf{b}_i = thresold(F(I_i; \Omega), 0.5) \end{aligned} \tag{3.1}$$

where $\mathcal{L}$ is an appropriate loss function; $\|\mathbf{b}_i - F(I_i; \Omega)\|_2^2$ measures the quantization error of between the DCNN activation $F(I_i; \Omega)$ and the binary code $\mathbf{b}_i$; λ is the coefficient controlling the quantization error; $threshold(v, t)$ is a thresholding function at t, and it equals to 1 if $v \geq t$, 0 otherwise; F is a composition of $n+1$ non-linear projection functions parameterized by Ω:

$$F(I, \Omega) = f_{\text{DBE}}(f_n(\cdots f_2(f_1(I; \omega_1); \omega_2) \cdots ; \omega_n)\omega_{\text{DBE}}), \tag{3.2}$$

where the inner n nonlinear projections composition denotes the n-layer DCNN; $f_{\text{DBE}}(\cdot; \omega_{\text{DBE}})$ is the Direct Binary Embedding layer appended to the DCNN. The binary code $\mathbf{b}_i$ in Eq. 3.1 makes it difficult to optimize via regular DCNN inference. We relax Eq. 3.1 to the following form where stochastic gradient descent is feasible:

$$\min_{W,F} \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{L}(\mathbf{W}^\top F(I_i; \Omega), y_i) + \lambda ||2F(I_i; \Omega) - \mathbf{1}| - \mathbf{1}|^2 \right) \tag{3.3}$$

As proved by [92], the quantization loss $||2F(I_i; \Omega) - \mathbf{1}| - \mathbf{1}|^2$ in Eq. 3.3 is an upper bound of that in Eq. 3.1, making Eq. 3.3 an appropriate relaxation and much easier to optimize.

Several studies such as [92] share the similar idea of encouraging the fully-connected layer representation to be binary codes by using hyperbolic tangent (tanh) activation. Since it is desirable to learn binary code $\mathbf{B} = \{0, +1\}^{N \times L}$, we propose to concatenate the ReLU (rectified linear unit) nonlinearity with the tanh nonlinearity. Formally, we define DBE layer

(shown in Figure 3.1):

$$\mathbf{Z} = f_{\text{DBE}}(\mathbf{X}) = \tanh(\text{ReLU}(\text{BN}(\mathbf{X}\mathbf{W}_{\text{DBE}} + b_{\text{DBE}}))) \tag{3.4}$$

where $\mathbf{X} = f_n(\cdots f_2(f_1(\mathbf{I}; \omega_1); \omega_2) \cdots; \omega_n) \in \mathbb{R}^{N \times d}$ is the activation of n-layer DCNN;

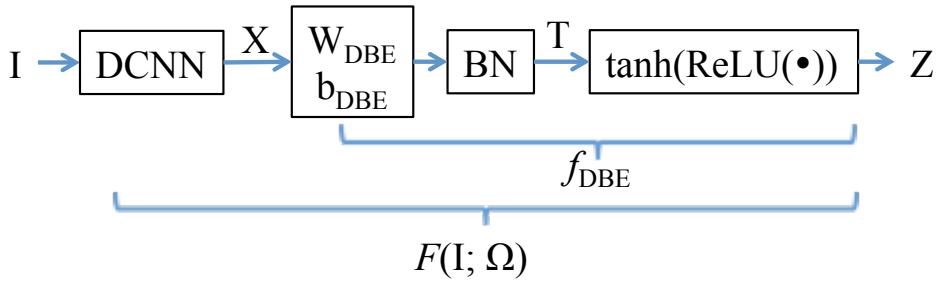


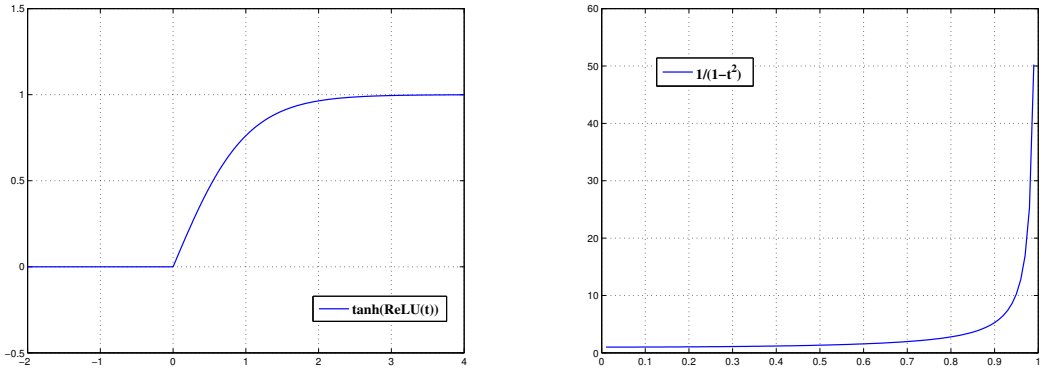
Figure 3.1: The framework of DBE and outputs of different projections

$\mathbf{Z} = f_{\text{DBE}}(\mathbf{X}) \in \mathbb{R}^{N \times L}$ is the binary-like activation of DBE layer; $\mathbf{T} = \text{BN}(\mathbf{X}\mathbf{W}_{\text{DBE}} + b_{\text{DBE}})$ is the activation after linear projection and batch normalization but prior to ReLU and tanh; $\mathbf{W}_{\text{DBE}} \in \mathbb{R}^{d \times L}$ is a linear projection, b_{DBE} is the bias; $\text{BN}(\cdot)$ is the batch normalization. And its activation is plotted in Figure 3.2a. The benefit of DBE layer approximating binary code is three-fold:

1. batch normalization mitigates training with saturating nonlinearity such as tanh [31], and potentially promotes more effective binary representation.
2. ReLU activation is sparse [18] and learns bit '0' inherently.
3. tanh activation bounds the ramping of ReLU activation and learns bit '1' effectively without jeopardizing the sparsity of ReLU.

Furthermore, DBE layer learns activation that approximates binary code statistically well. Consider random sampling t from $\mathbf{T}$, and assume it follows a distribution denoted by $p_T(t)$. Consequently the distribution of the DBE layer activation $z = f_{\text{DBE}}(t)$, and it follows distribution $p_Z(z)$, written as:

$$p_Z(z) = p_T(f_{\text{DBE}}^{-1}(z)) \left| \frac{1}{f'_{\text{DBE}}(f_{\text{DBE}}^{-1}(z))} \right| \tag{3.5}$$



(a) activation **(b)** The PDF for positive input

Figure 3.2: tanh(ReLU(·)) activation and its PDF for positive input

Eq. 3.5 holds since f_{DBE} is a monotonic and differentiable function. Since it is also positive when z is positive, thus we have:

$$p_Z(z) = p_X(f_{\text{DBE}}^{-1}(z))\frac{1}{1 - f_{\text{DBE}}^{-1}(z)^2}, \quad f_{\text{DBE}}^{-1}(z) = t > 0. \tag{3.6}$$

$p_T(f_{\text{DBE}}^{-1}(z))$ in Eq. 3.6 is equivalent to $p_T(t)$; $\frac{1}{1-f_{\text{DBE}}^{-1}(z)^2}$ grows sharply towards the discrete value $\{+1\}$ for any positive response z, as is plotted in Figure 3.2b. This suggests that the DBE layer enforces that the learned embedding z are assigned to $\{+1\}$ with large probability as long as z is positive. Conclusively DBE layer f_{DBE} can effectively approximate binary code. Eventually we choose to optimize Eq. 3.3 without the quantization error and replace the binary code $\mathbf{b}_i$ with DBE layer activation directly. Eq. 3.3 can thus be rewritten as:

$$\begin{aligned} &\min_{\mathbf{W},F} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathbf{W}^\top F(I_i;\Omega), y_i) \\ &\text{s.t. } F(\mathbf{I},\Omega) = f_{\text{DBE}}(f_n(\cdots f_2(f_1(\mathbf{I};\omega_1);\omega_2)\cdots;\omega_n)\omega_{\text{DBE}}) \end{aligned} \tag{3.7}$$

The inference of DBE is the same as canonical DCNN models via stochastic gradient descent (SGD).

3.3.2 Multiclass Image Classification

Majority of DCNNs are trained via multiclass classification using softmax cross entropy as the loss function. Following this paradigm, Eq. 3.7 can be instantiated as:

$$\begin{aligned} \min_{\mathbf{W},F} \; & -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C}\mathbb{1}(y_i)\log\frac{e^{\mathbf{w}_k^\top F(I_i;\Omega)}}{\sum_{j=1}^{C}e^{\mathbf{w}_j^\top F(I_i;\Omega)}} \\ \text{s.t. } & F(\mathbf{I},\Omega) = f_{\text{DBE}}(f_n(\cdots f_2(f_1(\mathbf{I};\omega_1);\omega_2)\cdots;\omega_n)\omega_{\text{DBE}}) \end{aligned} \tag{3.8}$$

where C is the number of categories; $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_C]$ and $\mathbf{w}_k,\ k = 1, \ldots, C$ is the weight of the classifier for category k; y_i is the label for image sample I, and $\mathbb{1}(y_i)$ an indicator function representing the probability distribution for label y_i. Essentially Eq. 3.8 aims to minimize the difference between the probability distribution of ground truth label and prediction.

3.3.3 Multilabel Image Classification

More often a real-world image is associated with multiple objects belonging to different categories. A natural formulation of optimization problem for multilabel classification is extending the multiclass softmax cross entropy in Eq. 3.8 to multilabel cross entropy. Indeed softmax cross entropy captures the co-occurrence dependencies among labels, one cannot ignore the independence of each individual labels. For instance, 'fork' and 'spoon' usually co-exist in an image as they are associated with super-concept 'dining'. But occasionally a 'laptop' can be placed randomly on the dining table where there are also 'fork' and 'spoon' in the image as well. Consequently, we propose to optimize a joint cross entropy by incorporating weighted binary sigmoid cross entropy, which models each label independently, to softmax cross entropy. Eq. 3.7 can therefore be instantiated as:

$$\begin{aligned} \min_{\mathbf{W},F} \; & -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c_+}\frac{1}{c_+}\log\frac{e^{\mathbf{w}_j^\top F(I_i;\Omega)}}{\sum_{p=1}^{C}e^{\mathbf{w}_p^\top F(I_i;\Omega)}} - \nu\frac{1}{N}\sum_{i=1}^{N}\sum_{p=1}^{C}\Bigg[\rho\mathbb{1}(y_i)\log\frac{1}{1+e^{\mathbf{w}_p^\top F(I_i;\Omega)}} \\ & +(1-\mathbb{1}(y_i))\log\frac{e^{\mathbf{w}_p^\top F(I_i;\Omega)}}{1+e^{\mathbf{w}_p^\top F(I_i;\Omega)}}\Bigg] \\ \text{s.t. } & F(\mathbf{I};\Omega) = f_{\text{DBE}}(f_n(\cdots f_2(f_1(\mathbf{I};\omega_1);\omega_2)\cdots;\omega_n)\omega_{\text{DBE}}) \end{aligned} \tag{3.9}$$

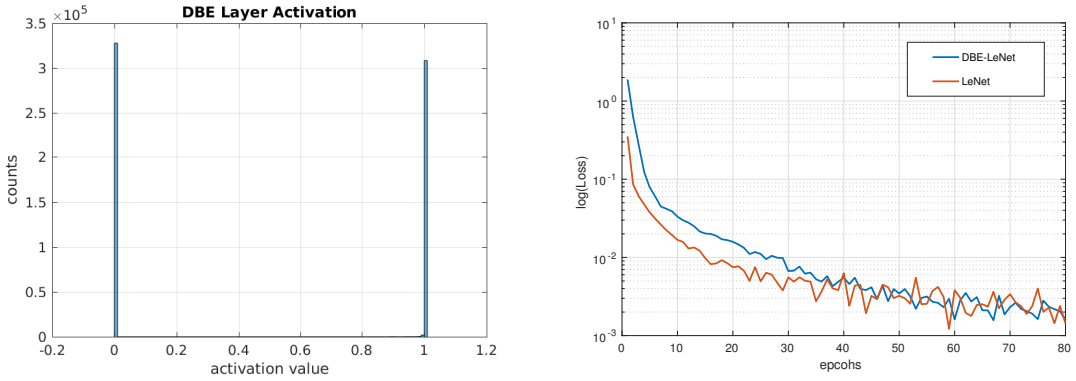
where c_+ is the number of positive labels for each image; ν is the coefficient controlling the numerical balance between softmax cross entropy and binary sigmoid cross entropy; ρ is the coefficient penalizing the loss for predicting positive labels incorrectly.

3.3.4 Toy Example: LeNet with MNIST Dataset

In order to demonstrate the effectiveness of DBE layer, we use LeNet as a simple example of DCNN. We add DBE layer to the last fully connected layer of LeNet and learn binary representation for MNIST dataset. **MNIST** dataset [43] contains 70K hand-written digits of 28×28 pixel size, ranging from '0' to '9'. The dataset is split into a 60K training set (including a 5K validation set) and a 10K test set[1]. We enhance the original LeNet with more convolutional kernels (16 kernels and 32 kernels on the first and second layer, respectively, all with size 3×3). We train the LeNet with DBE layer on the training set and evaluate the quality of learned binary representation on the test set. Figure 3.3a demonstrates the histogram of activation from DBE. Clearly DBE layer learns a representation approximating binary code effectively (51.1% of DBE activation less than 0.01, 48.6% greater than 0.99 and only 0.3% in between). We evaluate the quality of binary code learned by DBE qualitatively by comparing the classification accuracy on the test set with the state-of-the-art hashing algorithm. In order to demonstrate the effectiveness of DBE, we also compare with different λ in Eq. 3.3 for the purpose of showing that quantization error is not necessary anymore to learn high-quality binary representation. From Table 3.2 we can see that with the increase of λ in Eq. 3.3, the testing accuracy decreases. Due to the effectiveness of DBE layer, quantization error does not contribute to the binary code learning. Following the evaluation protocol of previous works [75], linear-SVM [15] is used as the classifier on all compared methods for fair comparison (including continuous LeNet representation). The classification accuracy on the test set is reported in Table 3.1.

The convergence of training DBE-LeNet is reported in Figure 3.3b. Due to the saturating tanh activation, the gradient is slightly more difficult to propagate through the network. Eventually the convergence reaches the same level.

[1] http://yann.lecun.com/exdb/mnist/



(a) (b)

Figure 3.3: The qualitatively results of DBE-LeNet: (a)The histogram of DBE layer activation; (b)The convergence of the original LeNet and with DBE trained on MNIST

Table 3.1: The comparison of the testing accuracy on MNIST. Code-length for all hashing algorithms is 64-bit. LeNet feature (1000-d continuous vectors) is used for SDH and FastHash.

Method	LeNet [43]	DBE-LeNet	SDH [75]	FastHash [45]
testing acc(%)	99.34	99.34	99.14	98.62

Table 3.2: The impact on quantization error coefficient λ

λ	0	1e-4	1e-3	1e-2	1e-1
testing acc(%)	99.34	99.34	99.30	99.26	99.01

3.4 Experiments

We evaluate the proposed DBE layer with the deep residual network (ResNet). We choose to append DBE layer to the state-of-the-art DCNN, 50-layer Residual Network (ResNet-50) [29] to learn high-quality binary representation for image sets. For the multilabel experiments, we set $\nu = 2$ and $\rho = 5$ through extensive empirical study.

3.4.1 Dataset

CIFAR-10 dataset [37] contains 60K color images (size 32×32) with each image containing a natural object. There are 10 categories of objects in total, with each category containing 6K images. The dataset is randomly split into a 50K training set and a 10K testing set. For

traditional image hashing algorithms, we provide 512-D GIST [63] feature; for end-to-end deep hashing algorithms, we use raw images as input directly. **Microsoft COCO 2014 (COCO)** [48] is a dataset for image recognition, segmentation and captioning. It contains a training set of 83K images with 605K annotations and a validation set of 40K images with 292K annotations. There are totally 80 categories of annotations. We treat annotations as labels for images. On average each image contains 7.3 labels. Since images in COCO are color images with various sizes, we resize them to 224×224.

3.4.2 Object Classification

To evaluate the capability of mulitclass object classification, we compare DBE with several state-of-the-art supervised approaches including FastHash [45], SDH [75], CCA-ITQ [20] and deep method DLBHC [47]. The ResNet-50 features are also included in the comparison. The code-length of binary code from all the hashing methods is 64 bits. We use linear-SVM to evaluate the all the approaches on the classification task.

Table 3.3 shows the classification accuracy on test sets for the two datasets. The accuracy achieved by DBE matches that of the original continuous ResNet-50 features. DBE improves the state-of-the-art traditional methods and end-to-end approaches by 28.6% and 5.6%, respectively. And it achieves the same performance as that of the original ResNet. This demonstrates 1) DBE's superior capability of preserving the rich semantic information extracted by ResNet, 2) there exists great redundancy in the original ResNet features.

Table 3.3: The testing accuracy of different methods on CIFAR-10 dataset. All binary representations have code-length of 64 bits.

Methods	Testing Accuracy (%)
CCA-ITQ [20]	56.34
FastHash [45]	57.82
SDH [75]	67.73
DLBHC [47]	86.73
ResNet [29]	**92.38**
DBE (ours)	<u>92.35</u>

Furthermore we also provide the classification accuracy on CIFAR-10 with respect to different code lengths in Table 3.4. From the table we can conclude that DBE learns high-quality binary representation consistently.

Table 3.4: Classification accuracy of DBE on CIFAR-10 dataset across different code lengths

Code length (bits)	16	32	48	64	128
testing acc(%)	91.63	92.04	92.20	92.35	92.36

3.4.3 Image Retrieval

Natural Object Retrieval

The CIFAR-10 dataset is used to evaluate the proposed DBE on natural object retrieval task. We choose to compare with state-of-the-art image hashing algorithms including both traditional hashing methods: CCA-ITQ [20], FastHash [45], and end-to-end deep hashing methods: DSH [51], DSRH [87]. For the experimental settings, we randomly select 100 images per category and obtain a query set with 1K images. Mean average precision (mAP) is used as the evaluation metric. The comparison is reported in Table 3.5. The proposed DBE outperforms state-of-the-art by around 3%. It confirms that DBE is capable of preserving rich semantics extracted by the ResNet from original images and learning high-quality binary code for retrieval purpose.

Table 3.5: Comparison of mean average precision (mAP) on CIFAR-10

Code length (bits)	12	24	36	48
CCA-ITQ [20]	0.261	0.289	0.307	0.310
FastHash [45]	0.286	0.324	0.371	0.382
SDH [75]	0.342	0.397	0.411	0.435
DSH [51]	0.616	0.651	0.661	0.676
DSRH [87]	0.792	0.794	0.792	0.792
DLBHC [47]	0.892	0.895	0.897	0.897
DBE (ours)	**0.912**	**0.924**	**0.926**	**0.927**

Multilabel Image Retrieval

COCO dataset is used for multilabel image retrieval task. Considering the large number of labels in COCO, we compare DBE with several cross modal hashing and quantization algorithms. Studies have shown that cross-modal hashing improves unimodal methods by leveraging semantic information of text/label modality [69, 32]. We choose to compare with CMFH [13] and CCA-ACQ [32]. Furthermore we also include traditional hashing method CCA-ITQ [20] and end-to-end approach DHN [92]. Following the experiment protocols in [32], 1000 images are randomly sampled from validation set for query and the training set is used for database for retrieval. And AlexNet [39] feature is used as input for algorithms that are not end-to-end, and raw images are used for end-to-end deep hashing algorithms. Due to the multilabel nature of COCO, we consider the true neighbors of a query image as the retrieved images sharing at least one labels with the query. Similar to natural object retrieval, mean average precision (mAP) is used as evaluation metric.

Table 3.6: Comparison of mean average precision (mAP) on COCO.

Code length (bits)	16	24	32	48	64
CCA-ITQ [20]	0.477	0.481	0.485	0.490	0.494
CMFH [13]	0.462	0.476	0.484	0.497	0.505
CCA-ACQ [32]	0.483	0.500	0.504	0.515	0.520
DHN [92]	0.507	0.539	0.550	0.559	0.570
DBE (ours)	**0.623**	**0.657**	**0.670**	**0.692**	**0.716**

3.4.4 Multilabel Image Annotation

We generate prediction of labels for each image in validation set based on K highest ranked labels and compare to the ground truth labels. The overall precision (O-P), recall (O-C), and F1-score (O-F1) of the prediction are used as evaluation metrics. Formally they are defined as:

$$\text{O-P} = \frac{N_{CP}}{N_P}, \ \text{O-R} = \frac{N_{CP}}{N_G}, \ \text{O-F1} = 2\frac{\text{O-P} \cdot \text{O-R}}{\text{O-P} + \text{O-R}} \tag{3.10}$$

where C is the number of annotations/labels; N_{CP} is the number of correctly predicted labels for validation set; N_P is the total number of predicted labels; N_G is the total number of ground truth labels for validation set.

We compare DBE with softmax, binary cross entropy and WARP [19], one of the state-of-the-art for multilabel image annotation. The performance comparison is summarized in Table 4.3 and we set $K = 3$ in the experiment. It can be observed that the binary representation learned by DBE achieves the best performance in terms of overall-F1 score. Due to its consideration of co-occurrence and independence of labels, DBE-joint cross entropy outperforms DBE-softmax and DBE-weighted binary cross entropy.

Table 3.7: Performance comparison on COCO for $K = 3$. The code length for all the DBE methods is 64-bit.

Method	O-P	O-R	O-F1
WARP [19]	**59.8**	61.4	60.6
DBE-Softmax	59.1	62.1	60.3
DBE-weighted binary cross entropy	57.1	60.8	58.9
DBE-joint cross entropy	59.5	**62.7**	**61.1**

3.4.5 The Impact of DCNN Structure

Similar to most deep hashing algorithms, DBE also preserves semantics from DCNN. Consequently the structure of DCNNs influences the quality of binary code significantly. We compare with the state-of-the-art DLBHC [47] and the DCNN it uses: AlexNet [39], which the upper bound in this comparison. Since DLBHC uses AlexNet, we also use AlexNet in our DBE. CIFAR-10 dataset is used. According to results reported in Table 3.8, DBE achieves higher accuracy than DLBHC, i.e., DBE learns more semantic and discriminative binary representation.

Table 3.8: The comparison of the classification accuracy on the test set of CIFAR-10. Code-length for all binary algorithms is 48-bit.

Method	AlexNet [39]	DBE-AlexNet	DLBHC [47]
testing acc(%)	89.20 (upper bound)	88.52	86.73

3.5 Conclusion

We proposed a novel approach to learn binary representation for images in an end-to-end fashion. By using a Direct Binary Embedding layer, we are able to approximate binary code directly in DCNN. Different from existing works, DBE learns high quality binary representation for images without quantization error as a regularization. Extensive experiments on two large-scale datasets demonstrate the effectiveness superiority of DBE over state-of-the-art on several computer vision tasks including object recognition, image retrieval and multilabel image annotation.

Chapter 4

Learning Binary Representation with Discriminative Cross-View Hashing

A version of this chapter was originally published by Liu Liu, Hairong Qi:

Liu Liu, Hairong Qi. "Discriminative Cross-View Binary Representation Learning." *IEEE Winter Conference on Applications of Computer Vision* (WACV), 2018.

4.1 Abstract

Learning compact representation is vital and challenging for large scale multimedia data. Cross-view hashing, or cross-modal hashing has received more attention with exponentially growing availability of multimedia content. Most existing cross-view hashing algorithms emphasize the alignment of different views via their similarities to achieve effective cross-view similarity search. In this work, we propose an end-to-end method to learn semantic-preserving and discriminative binary representation, dubbed Discriminative Cross-View Hashing (DCVH), in light of learning multitasking binary representation for various tasks including cross-view retrieval, image-to-image retrieval, and image annotation/tagging. This is achieved by exploiting convolutional neural network (CNN) based nonlinear projection and multilabel classification for both images and texts simultaneously. In addition, we achieve effective continuous relaxation for discrete hashing functions without explicit quantization loss by using Direct Binary Embedding (DBE) layers. Finally we propose an effective view alignment via Hamming distance minimization, which is efficiently accomplished by bit-wise XOR operation. Extensive experiments on two image-text benchmark datasets demonstrate that DCVH outperforms state-of-the-art cross-view hashing algorithms as well as single-view image hashing algorithms. In addition, DCVH can provide competitive performance for image annotation/tagging.

4.2 Introduction

Representation learning provides key insights and understanding of visual content, and thus is vital to computer vision tasks. On one hand, due to the increasing availability of image content, image hashing methods have been proposed to learn compact binary hash codes for similarity search purpose. Usually image hashing methods aim to project images onto

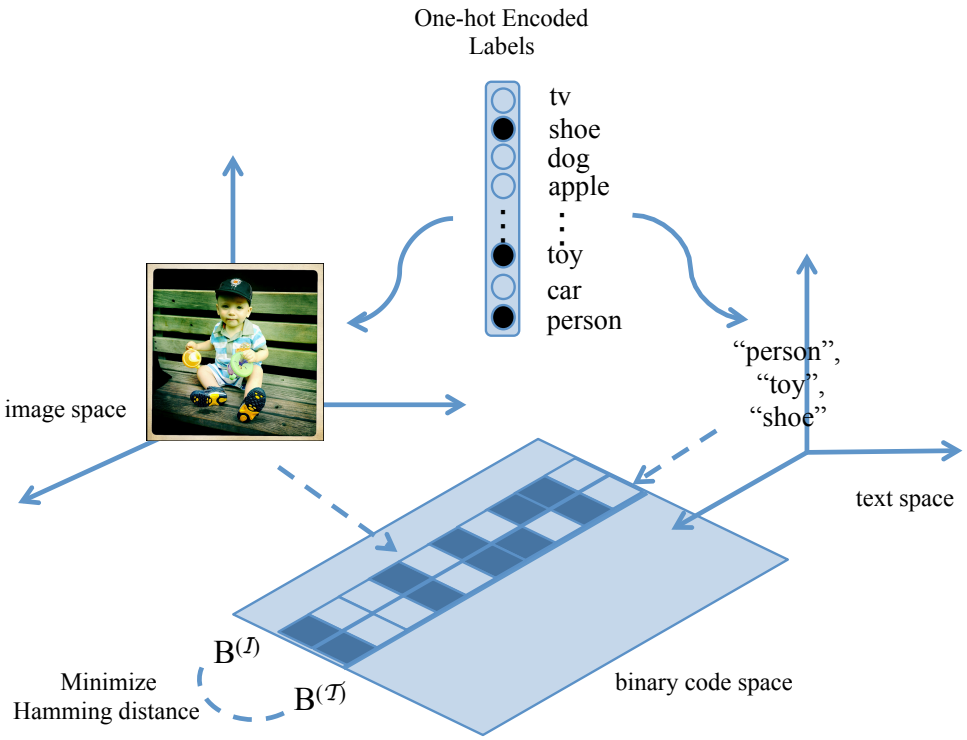


Figure 4.1: Rather than using cross-view similarities, DCVH learn discriminative view-specific binary representation via multilabel classification and align them by Hamming distance minimization via bit-wise XOR.

the Hamming space where semantic similarity in the original space is well preserved. This is often realized via pairwise similarity [51, 92] or triplet loss [94]. Several recent works also reveal that high-quality binary codes can be learned via classification task [52, 75, 53], and the learned codes provide great performance for both retrieval task and classification task. On the other hand, rapidly growing social media offer massive volumes of multimedia content as well, e.g., photo posts with textual tags on Flickr, tweets with pictures, etc. It is desired to perform efficient content understanding and analytics across different media modalities. Particularly, we are interested in understanding multimedia data involving images and textual information involving tags/annotations. To this end, cross-view hashing, or cross-modal hashing has been studied, and drawing great attention [41, 6, 13, 49, 32, 34, 8].

Cross-view hashing studies the heterogeneous relationship of different views of data (e.g. images and the associated textual information), attempting to map them into common Hamming space. This enables both inter- and intra-indexing of data from different views.

Previous methods mainly rely on a similarity matrix [34, 9, 41, 6] or provided affinities [49, 32] to delineate the cross-view relationship. However, representing such similarity is quite resource-intensive as its size tends to grow quadratically with the size of the training set; furthermore, the discriminative information that is useful to other tasks such as single-view image hashing and annotation is not well preserved by the similarities.

To address these problems, we propose Discriminative Cross-View Hashing (DCVH), an end-to-end approach, to learn semantic-preserving and *discriminative* binary representation. As illustrated in Figure 4.1, DCVH adopts multilabel classification directly to learn discriminative view-specific binary representation. Explicitly, a deep convolutional neural network [29] projects images into lower-dimensional latent feature space; for texts, DCVH uses pretrained GloVe [66] vectors to represent words in the texts. Then vector representation for each textual instance is formed by concatenating GloVe vectors. They are fed to a text-CNN [85], mapping text vectors into a common latent feature space of images. Meanwhile, a Direct Binary Embedding (DBE) layer [53] is employed in both deep CNN and text-CNN, enabling the learning of binary representations without explicit quantization loss. Finally, DCVH aligns different views by minimizing the Hamming distance between the view-specific binary representations directly. This is efficiently accomplished by bit-wise XOR operation. Extensive experiments are conducted to validate the proposed DCVH, and the results demonstrate that DCVH improves cross-view retrieval and single-view image retrieval tasks over state-of-the-art by a large margin. Meanwhile DCVH can also provide competitive performance on image annotation/tagging task, suggesting that DCVH learns multitasking binary representations. The contributions of this work can be outlined as follows.

1. we propose an effective end-to-end supervised cross-view binary representation learning algorithm: Discriminative Cross-View Hashing (DCVH), which learns semantic-preserving and discriminative binary representation for images and texts simultaneously.
2. we propose a novel approach of text embedding based on pretrained word vector representation and a text-CNN.

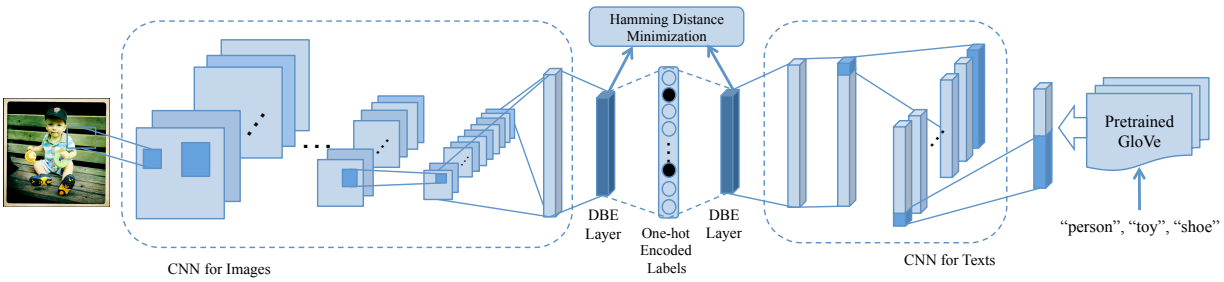


Figure 4.2: The overview architecture of the proposed Discriminative Cross-View Hashing (DCVH). Given multimedia data (images-texts), DCVH uses convolutional neural network (CNN) [29] to project images into binary representation; meanwhile DCVH uses pretrained GloVe [66] vectors to obtain text vector representation. Then text vectors are fed into a text-CNN [85] to generate binary representation. Unlike most methods that uses cross-view similarities, DCVH uses multilabel classification to embed raw images and texts into common binary feature space. Hammning distance minimization is adopted for view alignment purpose

3. we achieve effective view alignment, which directly minimizes the Hamming distance between the view-specific binary representation via bit-wise XOR operation, thanks to the inclusion of the DBE layer.

4. With DCVH, we can learn multitasking binary representation that can be used as high-quality hash code both for retrieval purpose, and compact image features for classification/annotation purpose.

The remainder of this paper is organized as follows. Section 4.3 discusses related work and their impact on the proposed method. Section 4.4 presents DCVH in details, including view-specific learning and view alignment. Section 4.5 reports the results of extensive experiments to validate DCVH on various tasks, i.e., cross-view retrieval, single-view image retrieval, and image annotation/tagging. Finally Section 4.6 summarizes this paper and presents concluding remarks.

4.3 Related Works

Our work is closely related to cross-view hashing or cross-modal hashing. Majority of cross-view hashing methods are based on hand-crafted features, and cannot provide end-to-end solutions. For instance, CCA [26] minimizes the distance between paired modalities

in a latent space on the condition that their norm is equal to one; CVH [41] is a cross-modal extension of Spectral Hashing [84]. CMSSH [84] maps multi-view data into common Hamming space by solving binary classification problem. CMFH [13] uses matrix factorization to solve the cross-modal hashing problem. ACQ[32] minimizes quantization error for different modalities alternatively while preserving data similarities. SePH [49] transforms affinities into global probabilities and learns view-invariant codes via KL-divergence. Recently several deep end-to-end cross-view hashing approaches have been proposed. For example, DVSH [8] employs CNN and LSTM for both images and sentences modality, respectively, while preserving the pairwise similarity via cosine max-margin loss. THN[9] proposes a hybrid deep architecture and use auxiliary datasets to learn cross-modal correlation for heterogeneous modalities using pairwise cross-entropy loss. DCMH [34] uses CNN and neural network (NN) for embedding images and texts separately and connect them with cross-view similarities. Similar to the hand-crafted feature based methods, these deep approaches also focus on cross-view similarities without considering view-specific discriminative information. This is different from our perspective on cross-view retrieval, which is presented in DCVH by using classification explicitly in separate views, and aligning the view by minimizing Hamming distance between the binary representations. This leads semantic-preserving and discriminative binary representation not only useful for cross-view retrieval, but also capable of single-view similarity search and image annotation/tagging tasks.

Our work is also related to image hashing via classification and multilabel image classification. Previous works such as SDH [75] and DBE [53] provide evidence that binary codes learned through classification tasks serve as strong hash code for retrieval tasks and image features for classification purpose, thanks to the discriminative information obtained via classification. This inspires us to adopt multilabel classification to learn discriminative binary representation for both images and texts, thus potentially competent for various tasks. Meanwhile, multilabel classification has attracted much attention. WARP [19] uses a ranking-based approach together with a CNN for image annotation. CNN-RNN [83] and DBE [53] suggest that binary cross entropy provides strong performance for classification task despite its simplicity. Consequently, we adopt binary cross entropy as the loss function.

Comparing to images, representation learning and classification for texts are solved differently. Since most natural language processing (NLP) problems deal with sequential data, e.g., sentences and documents, recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) are used [50] due to their capability of capturing long-term dynamics. Meanwhile, most cross-view hashing methods adopt bag-of-word (BoW) to represent textual information [49, 34, 41, 9]. Although simple, the rich semantic information of texts might not be well preserved. Alternatively, ACQ [32] represents texts with mean vectors of word vectors from word2vec features with linear transformation, leading to similar problem as well. Recent works suggest that CNNs are effective to solve NLP problems as well. For instance, THC [85] adopt one-dimensional CNN to learn binary representation of texts by using word features and position features. Meanwhile, most real-world images available through social media are associated with tags or annotations, where the sequential structure is not a strong as that of sentences. As suggested by fastText [36], texts can be conveniently modeled by a linear classifier with a hidden variable provided a lookup table for words. This inspires us to adopt a word embedding lookup table to embed words into vectors. 1-D CNN is then employed to learn binary representation, similar to that for images.

4.4 Proposed Algorithm

The proposed Discriminative Cross-View Hashing (DCVH) is presented in this section. It consists of three components: a deep structure that maps images into low-dimensional Hamming space; a lookup table for text vector representation followed by a text-CNN to embed textual information into common Hamming space; and a view alignment that minimizes the Hamming distance between corresponding image and text pair. The overview architecture of DCVH is illustrated in Figure 6.2.

In cross-view image hashing, a training set $\mathcal{S} = \{s_i\}_{i=1}^{N}$ consisting N instances is provided, where each instance in $\mathcal{S} = (\mathcal{I}, \mathcal{T})$ has two corresponding views: image-view $\mathcal{I} = \{I_i\}_{i=1}^{N}$ and text-view $\mathcal{T} = \{T_i\}_{i=1}^{N}$. We aim to generate semantic-preserving and discriminative binary representation $\mathbf{B}^{(\mathcal{S})} \in \{0, 1\}^{N \times D}$ for $(\mathcal{S})$ from the two views by the following hashing

functions

$$\mathbf{B}^{(\mathcal{S})} = thresold(F^{(\mathcal{S})}(\mathcal{S}; \Omega^{(\mathcal{S})}), 0.5),\ \mathcal{S} = \mathcal{I} \text{ or } \mathcal{T} \tag{4.1}$$
$$\text{s.t.} \quad threshold(v, t) = \begin{cases} 1, & v \geq t \\ 0, & \text{otherwise} \end{cases}$$

where $F^{(\mathcal{S})}(\cdot; \Omega^{(\mathcal{S})})$ is nonlinear projections parameterized by $\Omega^{(\mathcal{S})}$ for image-view $\mathcal{S} = \mathcal{I}$ and text-view $\mathcal{S} = \mathcal{T}$.

Discriminative information that could be useful for other tasks might not be well preserved via similarity-based supervision. Therefore, we adopt the paradigm of classification-based binary code learning [75, 53], and use textual information to obtain labels to classify both images and texts. One direct approach of generating the labels is to encode the text into one-hot labels according to whether a tag or annotation for an instance appears or not. And the label information is denoted as $\mathcal{Y} = \{y_i\}_{i=1}^{N}$.

Proper choices of nonlinear projection $F^{(\mathcal{I})}$ and $F^{(\mathcal{T})}$ would facilitate the learning of high-quality binary representation significantly. In this work, we choose CNN based projections for both image-view and text-view. Specifically, a deep CNN (e.g., ResNet [29]) concatenated with a Direct Binary Embedding (DBE) [53] layer for images; and a text-CNN [85] concatenated with a DBE layer for textual information are used. The DBE layer, as illustrated in Figure 4.3, learns a continuous binary-like representation $\mathbf{Z}^{(\mathcal{S})}$ that approximates the discrete 0-1 binary code well, i.e., $\mathbf{Z}^{(\mathcal{S})} \approx \mathbf{B}^{(\mathcal{S})}$. This effectively eliminates the need of quantization loss, originally commonly used by hashing methods [75, 32, 34, 92].

For texts, we employ a 2-conv layer text-CNN [85] with 1-D convolution, as demonstrated in Figure 6.2. Given vector embedding for textual information (e.g., concatenated GloVe vectors), text-CNN uses kernels with the same size as that of a GloVe vector on the first layer, and kernels with the same size as the output of the first conv layer output. Since we do not consider the sequential structure among texts, the stride size is also the same as the GloVe vector. There are 1,000 kernels on both convolutional layers and the second convolutional layer outputs 1-D vector directly. Then another fully-connected layer and a

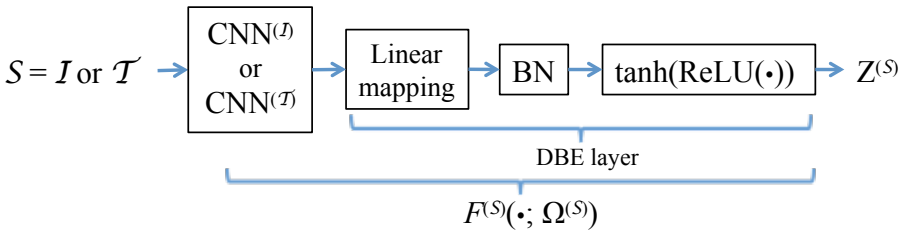


Figure 4.3: Direct Binary Embedding (DBE) layer architecture and how it is concatenated to a convolutional neural network (CNN) for images or texts. CNN$^{(\mathcal{I})}$ and CNN$^{(\mathcal{T})}$ are the corresponding CNN for images and text. DBE layer uses a linear transform and batch normalization (BN) with compound nonlinearity: rectified linear unit (ReLU) and tanh, to transform CNN activation to binary-like latent feature space $Z^{(\mathcal{S})}$

DBE-layer follow to embed the texts into a common latent space where image DBE features reside.

4.4.1 View-Specific Binary Representation Learning

Multilabel classification is used to learn binary representation for both images and texts. We choose binary sigmoid cross entropy as the loss function for both views. Linear classifiers $\mathbf{W}^{(\mathcal{S})}$, $\mathcal{S} = \mathcal{I}$ or $\mathcal{T}$, are used to classify the binary representation $\mathbf{B}^{(\mathcal{S})}$. Given the labels $\mathcal{Y} = \{y_i\}_{i=1}^N$, we have the following view-specific optimization problem:

$$\min_{\mathbf{W}^{(\mathcal{S})}, \Omega^{(\mathcal{S})}} L^{(\mathcal{S})}(\mathbf{W}^{(\mathcal{S})}, F^{(\mathcal{S})}(\mathcal{S}; \Omega^{(\mathcal{S})})) = \tag{4.2}$$

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{p=1}^{C}\left[\mathbb{1}(y_i)\log\frac{1}{1+e^{\mathbf{w}_p^{(\mathcal{S})\top}\mathbf{b}_i^{(\mathcal{S})}}} + (1-\mathbb{1}(y_i))\log\frac{e^{\mathbf{w}_p^{(\mathcal{S})\top}\mathbf{b}_i^{(\mathcal{S})}}}{1+e^{\mathbf{w}_p^{(\mathcal{S})\top}\mathbf{b}_i^{(\mathcal{S})}}}\right], \ \mathcal{S} = \mathcal{I} \text{ or } \mathcal{T}$$

$$\text{s.t.} \quad \mathbf{b}_i^{(\mathcal{S})} = threshold(F^{(\mathcal{S})}(s_i; \Omega^{(\mathcal{S})}), 0.5)$$

where C is the number of categories; $\mathbf{W}^{(\mathcal{S})} = [\mathbf{w}_1^{(\mathcal{S})}, \ldots, \mathbf{w}_C^{(\mathcal{S})}]$ and $\mathbf{w}_k^{(\mathcal{S})}$, $k = 1, \ldots, C$ is the weight of the classifier for category k; $\mathbb{1}(y_i)$ an indicator function representing the probability distribution for label y_i.

Direct optimizing Eq. 4.2 is difficult due to the discrete characteristics of $threshold(\cdot, \cdot)$. Meanwhile, since DBE layer approximates binary code well, Eq. 4.2 can be relaxed to the

following form without quantization loss:

$$\min_{\mathbf{W}^{(\mathcal{S})},\Omega^{(\mathcal{S})}} L^{(\mathcal{S})}(\mathbf{W}^{(\mathcal{S})}, F^{(\mathcal{S})}(\mathcal{S};\Omega^{(\mathcal{S})})) \approx \tag{4.3}$$
$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{p=1}^{C}\left[\mathbb{1}(y_i)\log\frac{1}{1+e^{\mathbf{w}_p^{(\mathcal{S})\top}F^{(\mathcal{S})}(s_i;\Omega^{(\mathcal{S})})}} + (1-\mathbb{1}(y_i))\log\frac{e^{\mathbf{w}_p^{(\mathcal{S})\top}F^{(\mathcal{S})}(s_i;\Omega^{(\mathcal{S})})}}{1+e^{\mathbf{w}_p^{(\mathcal{S})\top}F^{(\mathcal{S})}(s_i;\Omega^{(\mathcal{S})})}}\right],\ \mathcal{S}=\mathcal{I} \text{ or } \mathcal{T}$$

Eq. 4.3 suggests that the learning of the nonlinear projections and the classifiers are optimized in the end-to-end fashion.

4.4.2 View Alignment

For the purpose of effective cross-view indexing, it is necessary to align the learned binary representations from the two views. In order to do so, we propose to directly minimize the distance between learned binary representations. As a common distance metric for binary codes, Hamming distance can be conveniently expressed as *XOR* operation between two codes. Therefore, we attempt to minimize the Hamming distance between two corresponding binary representation $\mathbf{B}^{\mathcal{I}}$ and $\mathbf{B}^{\mathcal{T}}$ in order to achieve effective view alignment:

$$\begin{aligned}\min_{\Omega^{(\mathcal{I})},\Omega^{(\mathcal{T})}} & J_{\mathcal{I},\mathcal{T}}(F^{(\mathcal{I})}(\mathcal{I};\Omega^{(\mathcal{I})}), F^{(\mathcal{T})}(\mathcal{T};\Omega^{(\mathcal{T})})) \\ =&\frac{1}{ND}\sum \mathbf{B}^{\mathcal{I}}\oplus\mathbf{B}^{\mathcal{T}} \\ =&\frac{1}{ND}\sum \mathbf{B}^{\mathcal{I}}\odot\overline{\mathbf{B}}^{\mathcal{T}}+\overline{\mathbf{B}}^{\mathcal{I}}\odot\mathbf{B}^{\mathcal{T}} \\ &\text{s.t. } \mathbf{B}^{(\mathcal{S})}=thresold(F^{(\mathcal{S})}(\mathcal{S};\Omega^{(\mathcal{S})}),0.5),\ \mathcal{S}=\mathcal{I} \text{ or } \mathcal{T}\end{aligned} \tag{4.4}$$

where $\oplus$ is bit-wise XOR; $\odot$ is Hadamart multiplication or element-wise product. Since $\mathbf{B}^{(\mathcal{I})}$ and $\mathbf{B}^{(\mathcal{T})}$ are both binary, $\odot$ is equivalent to bit-wise AND. Similar to Section 4.4.1, Eq. 4.4 can be relaxed as:

$$\min_{\Omega^{(\mathcal{I})},\Omega^{(\mathcal{T})}} J_{\mathcal{I},\mathcal{T}}(F^{(\mathcal{I})}(;\Omega^{(\mathcal{I})}), F^{(\mathcal{T})}(;\Omega^{(\mathcal{T})})) \approx \tag{4.5}$$
$$\frac{1}{ND}\sum_{i=1}^{N}\left\{F^{(\mathcal{I})}(I_i;\Omega^{(\mathcal{I})})(\mathbf{1}-F^{(\mathcal{T})}(T_i;\Omega^{(\mathcal{T})}))^{\top}+(\mathbf{1}-F^{(\mathcal{I})}(I_i;\Omega^{(\mathcal{I})}))F^{(\mathcal{T})}(T_i;\Omega^{(\mathcal{T})})^{\top}\right\}$$

4.4.3 Total Formulation and Algorithm

DCVH learns cross-view binary representations by combining the view-specific learning and view alignment. Then the final formulation is

$$\begin{aligned} \min_{\substack{\mathbf{W}^{(\mathcal{I})},\mathbf{W}^{(\mathcal{T})},\\ \Omega^{(\mathcal{I})},\Omega^{(\mathcal{T})}}} & (1-\lambda)\left(L^{(\mathcal{I})}+L^{(\mathcal{T})}\right)+\lambda J_{\mathcal{I},\mathcal{T}}, \\ \text{s.t.} \quad & \lambda \in (0,1] \end{aligned} \tag{4.6}$$

where λ is a hyperparameter introduced to control the degree of the view alignment. Intuitively, the higher λ, the more matching two views will be, but the learned binary representation will be less discriminative; and vice versa. The detailed discussion on λ is included in Section 4.5.5.

The training of DCVH is the same as that of regular CNN, where stochastic gradient descent (SGD) is used to iterate through mini-batches of training data. In order to accelerate the training process, $F^{(\mathcal{I})}(\cdot;\Omega^{(\mathcal{I})})$ and $F^{(\mathcal{T})}(\cdot;\Omega^{(\mathcal{T})})$ are pretrained separately before the optimization of Eq. 4.7. Formally, DCVH is presented in Algorithm 2.

For a test sample that is not in the training set, the binary representation can be obtained via Eq. 4.1. And retrieving from database can be efficiently performed by ranking the retrieved results according the Hamming distance; or using pre-defined Hamming radius to perform hash lookup. For tagging/annotation purpose, the predicted tags or annotations for the test sample image can be obtained from the top-k predictions by using the classifier $\mathbf{W}^{(\mathcal{I})}$, where k is the number of tags or annotations. This overlaps with the task of retrieving textual information given test sample images, but it is more accurate since usually a retrieval is considered successful as long as one tag/annotation matches, where multilabel classification requires top-k predication matches simultaneously.

4.4.4 Extension

Although DCVH is proposed in the discussion of cross-view data, it is easily extended to the circumstances where three or more views of data is available. If there are m views of data, i.e., $\mathcal{S} = (S_1, S_2, \ldots, S_m)$. A direct way to extend Eq. 4.7 is by considering mutual Hamming

Algorithm 2 The Inference of DCVH

Input: The training set $\mathcal{S}$ consisting of corresponding images $\mathcal{I}$, texts $\mathcal{T}$ and the one-hot labels $\mathcal{Y}$ obtained from texts.
Output: Parameters of nonlinear projections $F^{(\mathcal{I})}(\cdot;\Omega^{(\mathcal{I})})$, $F^{(\mathcal{T})}(\cdot;\Omega^{(\mathcal{T})})$, and linear classifier $\mathbf{W}^{(\mathcal{I})}, \mathbf{W}^{(\mathcal{T})}$.

Initialization: Initialize vector embedding of texts with pretrained GloVe embedding. Initialize learning rate with μ, mini-batch size with 64, and maximum iterations with ITER.
Pretraining: Pretrain the nonlinear projections $F^{(\mathcal{I})}(\cdot;\Omega^{(\mathcal{I})})$ with $\mathbf{W}^{(\mathcal{I})}$, and $F^{(\mathcal{T})}(\cdot;\Omega^{(\mathcal{T})})$ with $\mathbf{W}^{(\mathcal{T})}$ separately according to Eq. 4.3.
for ITER **do**
Update $\Omega^{(\mathcal{I})}$ by

$$\Omega^{(\mathcal{I})} \leftarrow \Omega^{(\mathcal{I})} - \mu \frac{\partial}{\partial \Omega^{(\mathcal{I})}} \left((1-\lambda)L^{(\mathcal{I})} + \lambda J\right)$$

Update $\Omega^{(\mathcal{T})}$ by

$$\Omega^{(\mathcal{T})} \leftarrow \Omega^{(\mathcal{T})} - \mu \frac{\partial}{\partial \Omega^{(\mathcal{T})}} \left((1-\lambda)L^{(\mathcal{T})} + \lambda J\right)$$

Update $\mathbf{W}^{(\mathcal{I})}$ by

$$\mathbf{W}^{(\mathcal{I})} \leftarrow \mathbf{W}^{(\mathcal{I})} - \mu(1-\lambda) \frac{\partial}{\partial \mathbf{W}^{(\mathcal{I})}} L^{(\mathcal{I})}$$

Update $\mathbf{W}^{(\mathcal{T})}$ by

$$\mathbf{W}^{(\mathcal{T})} \leftarrow \mathbf{W}^{(\mathcal{T})} - \mu(1-\lambda) \frac{\partial}{\partial \mathbf{W}^{(\mathcal{T})}} L^{(\mathcal{T})}$$

end for

distance minimization:

$$\min_{\mathbf{W}^{(S_i)}, \Omega^{(S_i)}, \forall i} (1-\lambda) \sum_{i=1}^{m} L^{(S_i)} + \lambda \sum_{i,j} J_{i,j}, \tag{4.7}$$
$$\text{s.t.} \quad \lambda \in (0, 1]$$

where $J_{i,j}$ is the Hamming distance between the binary representation of ith and jth view.

4.5 Experiments

4.5.1 Datasets

We evaluate the proposed DCVH on two image-text benchmark datasets: MS COCO [48] and MIRFLICKR [30].

MS COCO is a dataset for image recognition, segmentation and captioning. It contains a training set of 83K images and a validation set of 40K images. There are totally 80

categories of annotations with several annotations per image. Since images in COCO are color images with various sizes, we resize them to 224×224. The textual representation for DCVH are obtained by concatenating pretrained GloVe [66] vectors. And all the text vectors are zero-padded to the same size, resulting in a 6000-D vector per image. Text vectors for the compared algorithms are obtained from word2vec [60] vectors, as suggested by ACQ [32]. For the hand-crafted feature based algorithms, images are represented by AlexNet [39] activation features; raw images are used for end-to-end algorithms.

MIRFLICKR contains 25K color images originally collected from Flickr, and each image is associated with several textual tags. All the images are also resized to 224×224. Following the settings of DCMH [34], we remove the images with noisy textual information and without valid textual information. Similarly, the textual information for each image is represented by a 4200-D vector from pretrained GloVe embeddings for the proposed DCVH. For the compared methods, the texts for the comparing algorithms is represented as BoW vectors as suggested by them.

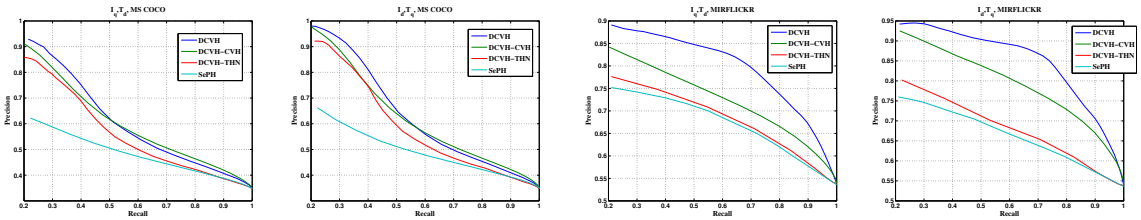
4.5.2 Experimental Settings and Protocols

Several state-of-the-art methods for cross-view hashing, supervised image hashing and image annotation are adopted as baselines to compare with DCVH. Since real-world images usually contain multiple tags or visual concepts, we consider a retrieval is successful when the retrieved item shares at least one concept with the query. We set hyperparameter $\lambda = 0.2$ of DCVH throughout the experiments. Learning rate μ is set to 0.0002 for pretraining, and 0.0001 with exponential decay for further training. We pretrain ResNet for images 10,000 iterations and text-CNN for texts 2,000 iterations; and further train them together with view alignment 10,000 iterations. DCVH is implemented in TensorFlow [1].

For the MS COCO dataset, we choose to compare with several state-of-the-art hand-crafted feature based algorithms including CVH [41], CMSSH [6], CMFH [13], SePH [49], ACQ [32]. We also compare with end-to-end approach. Since the code for most deep algorithms is not available, we adopt the pairwise cross-entropy loss provided in THN [9], and weighted average Hamming distance between different views in CVH [41] as the view alignment, and propose two variations of DCVH, denoted as DCVH-THN and DCVH-CVH.

Table 4.1: mAP values for the proposed DCVH and compared baselines on cross-view retrieval task with all the benchmark datasets. Results are provided for different code lengths.

Task	Method	MS COCO			MIRFLICKR		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Image→Text	CVH [41]	0.484	0.471	0.435	0.607	0.618	0.616
	CMSSH [6]	0.472	0.467	0.453	0.573	0.574	0.571
	CMFH [13]	0.486	0.517	0.545	0.586	0.584	0.584
	SePH [49]	0.543	0.551	0.557	0.672	0.677	0.679
	ACQ [32]	0.531	0.544	0.555	-	-	-
	DCMH [34]	-	-	-	0.713	0.720	0.730
	DCVH-THN	0.601	0.618	0.623	0.681	0.692	0.706
	DCVH-CVH	0.703	0.721	0.728	0.710	0.723	0.728
	DCVH	**0.710**	**0.728**	**0.733**	**0.715**	**0.745**	**0.769**
Text→Image	CVH [41]	0.480	0.467	0.432	0.603	0.604	0.602
	CMSSH [6]	0.465	0.454	0.446	0.572	0.573	0.570
	CMFH [13]	0.486	0.517	0.545	0.586	0.584	0.584
	SePH [49]	0.549	0.557	0.562	0.720	0.727	0.731
	ACQ [32]	0.521	0.546	0.562	-	-	-
	DCMH [34]	-	-	-	0.750	0.758	0.770
	DCVH-THN	0.619	0.631	0.645	0.719	0.734	0.743
	DCVH-CVH	0.728	0.749	0.753	0.738	0.754	0.773
	DCVH	**0.739**	**0.757**	**0.763**	**0.772**	**0.798**	**0.810**



(a) I-Q, T-D, MS COCO **(b)** I-D, T-Q, MS COCO **(c)** I-Q, T-D, MIRFLICKR **(d)** I-D, T-Q, MIRFLICKR

Figure 4.4: Comparison of precision-recall curve with code length of 64 bits on tasks: image query with text dataset (I-D, T-D) and text query with image dataset (I-D, T-D), on MS COCO ((a), (b)), and MIFLICKR ((c), (d)).

1,000 samples are randomly selected to form the query set and the rest is treated as the database for retrieval purpose and training. Note that DCVH and its variations are trained on the provided training set only. In addition, we compare with several state-of-the-art supervised image hashing algorithms for the purpose of showing DCVH is able to improve single-view image hashing performance as well. 5,000 images are randomly selected as the query set. We choose to compare with HashNet [10], DBE [53], DHN [92], KSH [56], ACQ, and CMFH.

For the MIRFLICKR dataset, we choose to compare with hand-crafted feature based algorithms: CVH, CMSSH, CMFH, SePH. We also compare with DCMH, which is an end-to-end approach. DCVH-THN and DCVH-CVH are included in the comparison as well. 2,000 samples are randomly picked as the query set and the rest are treated as database; meanwhile 5,000 randomly selected samples are used for training.

We evaluate the quality of the proposed DCVH via retrieval task and annotation task. Mean average precision (mAP) and precision-recall curve are adopted as evaluation metrics for Hamming ranking and hash lookup retrieval procedures, respectively. The prediction of tags or annotations for images can be obtained directly via the linear classifier $\mathbf{W}^{\mathcal{I}}$ of the proposed algorithm. Based on the K highest ranked prediction, the overall precision (O-P), recall (O-C) and F1-score (O-F1) are used as evaluation metrics:

$$\text{O-P} = \frac{N_{CP}}{N_P},\ \text{O-R} = \frac{N_{CP}}{N_G},\ \text{O-F1} = 2\frac{\text{O-P} \cdot \text{O-R}}{\text{O-P} + \text{O-R}} \tag{4.8}$$

where C is the number of tags/annotations; N_{CP} is the number of correctly predictions for validation set; N_P is the total number of predictions; N_G is the total number of ground truth for validation set.

4.5.3 Results of Cross-view Retrieval Task

Cross-view Retrieval

For DCVH and all the compared baselines, the cross-view retrieval performance based on Hamming ranking on all the datasets is reported in Table 4.1, including the task of image retrieval with text query and text retrieval with image query. We can observe that DCVH

generally outperforms the compared methods on the two benchmark datasets with various code lengths. For the MS COCO datasets, since DCMH does not provide results, we can observe that on image query retrieving from text database, DCVH output its two variations DCVH-THN and DCVH-CVH by 11% and 0.5%, respectively; on text query retrieving from image database, DCVH improves DCVH-THN and DCVH-CVH by 12% and 1% for all code lengths, respectively. DCVH also outperforms the hand-crafted feature based algorithms by at least 16% on both cross-view retrieval tasks. For the MIRFLICKR dataset, DCVH outperforms DCMH on image query retrieving from text database task by 0.2%, 2.5%, and 3.9% for code length of 16 bits, 32 bits, and 64 bits, respectively; on text query retrieving from image database task, DCVH improves DCMH by 2.2%, 4% and 4% for code length of 16 bits, 32 bits, and 64 bits, respectively. It also outperforms hand-crafted feature based algorithms such as SePH by from around 4% to 9% on both cross-view retrieval tasks over various code lengths.

Furthermore, we use hash lookup to compare DCVH with its two variations DCVH-THN and DCVH-CVH to validate the effectiveness of the view-alignment used in DCVH. SePH is also included into the comparison as the baseline. The comparison is summarized in Figure 4.4 in terms of precision-recall curve. For the MS COCO dataset, although DCVH obtains slightly lower precision for higher recall level on two cross-view retrieval tasks, it outperforms its variations and SePH in general. For the MIRFLICKR dataset, DCVH achieves the highest precision at all recall level comparing to other methods.

The comparison of mAP and precision-recall curve on the two benchmark datasets confirms the superiority of the proposed DCVH. As an end-to-end approach, DCVH not only captures the rich semantics from images using CNN, it also is able to extract textual information from pretrained GloVe vectors by using text-CNN. Furthermore, the comparison results of DCVH and its variations validates that the view alignment employed by DCVH is more effective.

Single-view Image retrieval

As suggested by ACQ [32], cross-view hashing can improve single-view similarity search. This is because the semantic information of the textual data is carries over to image view

Table 4.2: Comparison of mean average precision (mAP) on COCO for single-view image retrieval task

Code length	16 bits	32 bits	48 bits	64 bits
CMFH [13]	0.476	0.484	0.497	0.505
CCA-ACQ [32]	0.500	0.504	0.515	0.520
KSH [56]	0.521	0.534	0.534	0.536
DHN [92]	0.677	0.701	0.695	0.694
DBE [53]	0.623	0.670	0.692	0.716
HashNet [10]	0.687	0.718	0.730	0.736
DCVH	**0.721**	**0.748**	**0.757**	**0.761**

thanks to the view alignment. We compare DCVH with several state-of-the-art supervised image hashing algorithms, as well as with cross-view image hashing algorithms. Similar to the cross-view image hashing, the comparison result is reported in terms of mAP, as shown in Table 4.2. We can observe that DCVH improves HashNet by around 2.5% across various code lengths on MS COCO dataset. Similar to DCVH, DBE also learns binary representation via classification. DCVH outperforms DBE around 5% with different code lengths.

4.5.4 Image Annotation

We compare DCVH with several state-of-the-art multilabel image annotation algorithms including WARP [19] and DBE [53] on the MS COCO dataset. Note that the performance is evaluated on validation set of MS COCO, which involves 40K samples. Using overall-precision (O-P), overall-recall (O-R) and overall-F1 score (O-F1), the results are based on top-3 prediction of annotations, and are summarized in Table 4.3. From the table we can see that DCVH is able to provide competitive results for image annotation task. This suggests that despite the compromise for the view alignment, DCVH still manages to provide strong performance on image annotation/tagging task. Comparing to its variations, DCVH presents slightly improved performance, suggesting that our proposed view alignment causes minimal interference during the learning of discriminative information. This shows that the binary representation learned by DCVH can be used for different visual tasks, making DCVH a multitasking binary representation learning method.

Table 4.3: Performance comparison on MS COCO for image annotation task, compared on top-3 predictions.

Method	O-P	O-R	O-F1
WARP [19]	0.598	0.614	0.606
DBE [53] (64 bits)	0.595	**0.627**	**0.611**
DCVH-THN (64 bits)	0.596	0.615	0.605
DCVH-CVH (64 bits)	0.583	0.604	0.594
DCVH (16 bits)	0.546	0.563	0.554
DCVH (32 bits)	0.572	0.591	0.581
DCVH (64 bits)	**0.601**	0.617	0.609

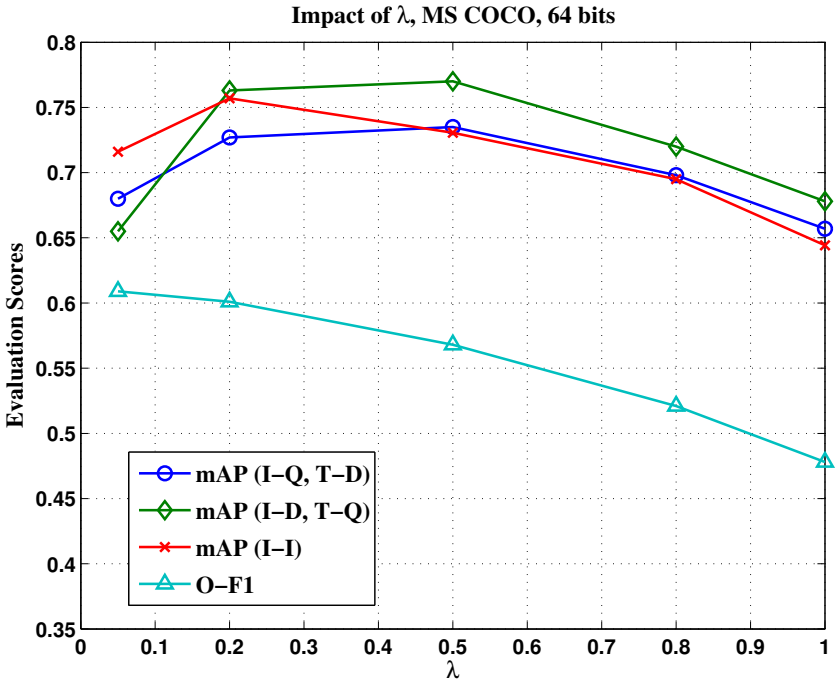


Figure 4.5: Impact of λ on DCVH, evaluated on several tasks including image query retrieving from text database (I-Q, T-D); text query retrieving from image database (I-D, T-Q); image query retrieving from image database (I-I); and image annotation. mAP is used as evaluation metric for retrieval tasks and overall F1 score (O-F1) is used for annotation task. The code length is set as 64 bits across the experiments.

4.5.5 Impact of Hyperparameter

In this section, we provide experimental analysis on the impact of hyperparameter λ. Generally λ controls the strength of view alignment and the discriminativeness of the learned binary representation. For the purpose of showing such characteristics, we set λ to different value (0.05, 0.2, 0.5, 0.8, 1.0) and evaluate its impact according to the performance of DCVH on cross-view retrieval, single-view image retrieval and image annotation tasks. All the experiments are conducted on MS COCO. Figure 4.5 summarizes the results w.r.t. various λ values.

It can be observed from Figure 4.5 that the cross-view retrieval performance dwindles when λ is either too large or too small. This is because when $\lambda \to 1$, the two views are strongly aligned while the discriminative information (supervision from labels) being very weak. Consequently the semantics from images and views cannot be well preserved in the binary representations. On the contrary, when $\lambda \to 0$, the semantics of the two views are not aligned well, leading to poor performance on retrieval across different views. Interestingly, the single-view image retrieval performance is enhanced when λ is near 0.2. We argue that the semantics from texts provides extra information for similarity search, although the image retrieval performance generally goes down with the increasing λ. Finally, we can see that image annotation is best performed by DCVH when λ is near 0. And its performance goes down more significantly especially when λ is greater than 0.2. By considering different tasks performed by DCVH, we choose to set $\lambda = 0.2$.

4.6 Conclusions

In this paper, we proposed a binary representation learning method: Discriminative Cross-View Hashing. It is an end-to-end approach for effective multimedia content understanding, and includes several novelties to learn high-quality binary representations. (a) It adopts multilabel classification to learn discriminative representation. (b) It employs pretrained GloVe vectors to obtain semantic text representation. (c) It uses deep architectures such as ResNet and text-CNN, together with Direct Binary Embedding layer to learn effective hashing functions, yielding high-quality binary codes (d) It exploit an effective view

alignment scheme, which uses bit-wise XOR operation for Hamming distance minimization purpose. Extensive experiments conducted on two benchmark datasets suggest that DCVH learns binary representation that provides superior performance on various computer vision tasks.

Chapter 5

Cross-Domain Image Hashing with Adversarial Learning

A version of this chapter was originally accepted by ICCV 2017 TASK-CV Workshop

5.1 Abstract

In this work, we study the problem of learning domain adapted binary representation for cross-domain images, aiming to bridge the gap between the labeled source domain and the unlabeled target domain. Recent development in image hashing for unsupervised domain adaptation shows the success of the adapted binary representation provides on tasks including unlabeled domain prediction and cross-domain retrieval. Usually hash codes are learned based on similarity information and the discriminative information is exploited inadequately; also the domains are compared implicitly by metric-based methods. To address these problems, we propose a novel algorithm that progressively learns binary representation via adversarial domain adaptation, dubbed progressive Adversarial Binary Representation (p-ABR). It has the following keys: first, discriminative binary representation is learned by progressive classification tasks using an ensemble of classifiers. This leads to higher-quality representation binary representation of various code-lengths simultaneously. Second, we achieve better domain adaptation by using adversarial learning, which matches domain distributions explicitly. Finally, we obtain effective hashing functions by using separate nonlinear projections including CNNs and Direct Binary Embedding (DBE) layers for different domains to respect the potential domain discrepancies. Extensive experiments are conducted on both standard and open set domain adaptation, and the results show that our method outperform previous state-of-the-art algorithms in different tasks.

5.2 Introduction

Learning binary representation is an efficient and pervasive solution for understanding large scale visual content [90, 75, 87, 21, 9, 34], especially considering the rapid growth of modern media such as social networks. Usually known as hashing, binary representation learning projects high-dimensional data onto the low-dimensional Hamming space while preserving semantic information of the original data. Most recent development in binary

representation learning, such as deep hashing [90, 34] are end-to-end approaches that employ deep convolutional neural networks (CNNs) [2, 29]. However, deep hashing often relies on supervised learning at the expense of large volume of manually labeled or annotated data, which generally is prohibitively expensive and time-consuming to obtain. Furthermore, a nontrivial yet rarely addressed problem is how to efficiently learn binary representation of various code-lengths. Due to different task requirements or evaluation purpose, it is often required to learn binary representation with various lengths. Most approaches ignore this and simply retraining/fine-tuning the model.

On the other hand, despite the success of deep learning especially in computer vision, the inevitable variances of models bring degradation to the performance when applied on datasets that have different distribution from the one which the models are trained in the first place. Although this is solvable thanks to the transferability of deep structure [88] or through fine-tuning, it is often impractical due to the lack of supervision (e.g., newly collected dataset not manually labeled/annotated yet) that is necessary for fine-tuning. Such non-negligible distributional difference known as domain shift hinders a better understanding of the unlabeled datasets. To address such domain shift, *domain adaptation* methods have been proposed. They generally identify such shift based on certain metrics and minimize it directly [89, 9, 79, 58], or implicitly align domains by domain-adversarial loss [16, 80]. However majority of works do not discuss the potential *semantic* shift or difference between different domains, i.e., their semantics are only partially overlapped.

To address the aforementioned problems, we propose a novel approach to learning discriminative and domain-invariant binary representation in a progressive way via adversarial domain adaptation, dubbed progressive Adversarial Binary Representation (p-ABR). Unlike most hashing algorithms that use similarity metric such as pairwise similarity and triplet loss, p-ABR learns discriminative binary representation directly via the classification task on the labeled domain, leading to high-quality multifunctional representation for retrieval and prediction tasks. In the meantime, by progressively learning representation of various code-length simultaneously, p-ABR utilizes the prediction made by shorter representation as strong prior evidence, and makes more accurate new prediction with longer representation. This leads to both overall high-quality representation and learning

representation of various code-lengths simultaneously. p-ABR achieves domain-invariant representation learning by matching domain distributions explicitly via adversarial domain adaptation [80]. This is different from most metric-based domain adaptation methods such as max mean discrepancy (MMD) [24] and central moment discrepancy (CMD) [89] that compare distribution implicitly. Finally, by using Direct Binary Embedding (DBE) [53] layers in both domain models, p-ABR achieves effective hashing functions without the usage of quantization loss, which is popular in hashing algorithms, to enforce the approximation of binary codes. Extensive experiment validates the proposed p-ABR: we consider both standard and open set domain adaptation scenarios and p-ABR provides state-of-the-art performance consistently.

5.3 Related Works

Recent advancement in image hashing suggests that learning discriminative binary representation is effective for both retrieval and classification task [75, 53]. Different from conventional image hashing that preserves pairwise similarities of original data, DBR preserves the label semantic information, learning multitasking binary representations for images. SDH [75] and CE-Bits [52] use L2 and hinge loss, as well as cross-entropy as classification loss criteria, but they are not end-to-end methods and rely on extracted features; DBE [53] takes advantage of deep CNN and learns efficient end-to-end hashing function without quantization loss.

Our work is also closely related to unsupervised discriminative domain adaptation. It aligns source and target domains so that the classifier trained on the source domain can predict in the target domain [79, 16]. THN [9] uses MMD [24] to align homogeneous domains. RTN [58] jointly learns adaptive classifiers and transferable features with MMD. Deep CORAL [79] directly minimizes the mean and covariance of latent features. Central moment discrepancy [89] matches the moments for each order and does not require distance or kernel computations. Meanwhile, adversarial adaptation employs a discriminator. While it tries to distinguish images w.r.t. their domains, the feature generators (CNNs) try to confuse the discriminator, leading to a minimax game [23]. Ganin [16] adopts adversarial learning

and proposes to confuse a domain classifier via a gradient reversal layer. ADDA [80] unifies adversarial domain adaptation approaches and learns separate CNNs for different domains. DAH [81] pioneered domain adapted image hashing. By mainly using pairwise similarity and a single CNN to map images from different domains to lower dimension, it uses MMD to couple the fully-connected layers and achieves state-of-the-art cross-domain retrieval and prediction in the unlabeled domain. In contrast, we use separate CNNs for different domains to respect the potential domain discrepancies even in lower hierarchical convolutional layers. ATI [65] discussed the open set domain adaptation problem and proposed to solve it by iteratively updating assignment and transformation between source and target domain. Note that the open set domain adaptation in our empirical study is different from ATI, as we assume that the source domain or the labeled domain is fully understood.

5.4 Proposed Algorithm

In the cross-domain discriminative binary representation learning problem, we intend to learn discriminative binary representation for the unlabeled domain $\mathcal{D}_u$ w.r.t. the labeled domain $\mathcal{D}_l$. Denote images of the labeled domain as $(\mathbf{I}^l, \mathbf{Y}^l) = \{(I_i^l, y_i^l)\}_{i=0}^{N-1}$, where $\mathbf{I}^l$ is the image set and $\mathbf{Y}^l \in \{0,1\}^{N\times C}$ are the corresponding one-hot encoded labels with C categories, and there are N labeled images in domain $\mathcal{D}_l$. Denote images from the unlabeled domain as $\mathbf{I}^u = \{I_i^u\}_{i=0}^{M-1}$, where there are M images in $\mathcal{D}_u$. We aim to learn binary representation $\mathbf{B}^l = \{\mathbf{b}_i^l\}_{i=0}^{N-1} \in \{0,1\}^{N\times L}$ and $\mathbf{B}^u = \{\mathbf{b}_i^u\}_{i=0}^{M-1} \in \{0,1\}^{M\times L}$ (L is the code length) for images from both $\mathcal{D}_l$ and $\mathcal{D}_u$, respectively.

Since discriminative binary representation can be obtained through classification task, usually we train a linear classifier to classify $\mathbf{B}^l$ given label information $\mathbf{Y}^l$; meanwhile, it is necessary to address the domain difference, for which the correspondence between domains is unavailable. Naturally, we formulate the following optimization problem

$$\begin{aligned} &\min_{\mathbf{W},H} \ \mathcal{L}_{\text{supervised}}(\mathbf{B}^l, \mathbf{Y}^l) + \mathcal{L}_{\text{unsupervised}}(\mathbf{B}^l, \mathbf{B}^u) \\ &\text{s.t. } \mathbf{B}^l = threshold(H^l(I^l;\theta^l), 0.5), \\ &\qquad \mathbf{B}^u = threshold(H^u(I^u;\theta^u), 0.5). \end{aligned} \tag{5.1}$$

where $\mathcal{L}_{\text{supervised}}$ is a supervised loss function for classification in the labeled domain, measuring differences between prediction posterior probability and ground truth distribution; $\mathcal{L}_{\text{unsupervised}}$ accounts for the distributional difference between the two domains; $threshold(\cdot, 0.5)$ is the thresholding operation at 0.5, i.e., it returns 1 when input is larger than 0.5, and 0 otherwise; H^l and H^u are the hashing function parameterized by θ^l and θ^u for domain $\mathcal{D}_l$ and $\mathcal{D}_u$, respectively. Unlike many domain adaptation methods using the same nonlinear projection such as CNN for different domains [81, 79], we use two separate CNNs in p-ABR to respect the potential discrepancies even in lower hierarchy convolutional layers. The overall architecture is illustrated in Fig. 5.1.

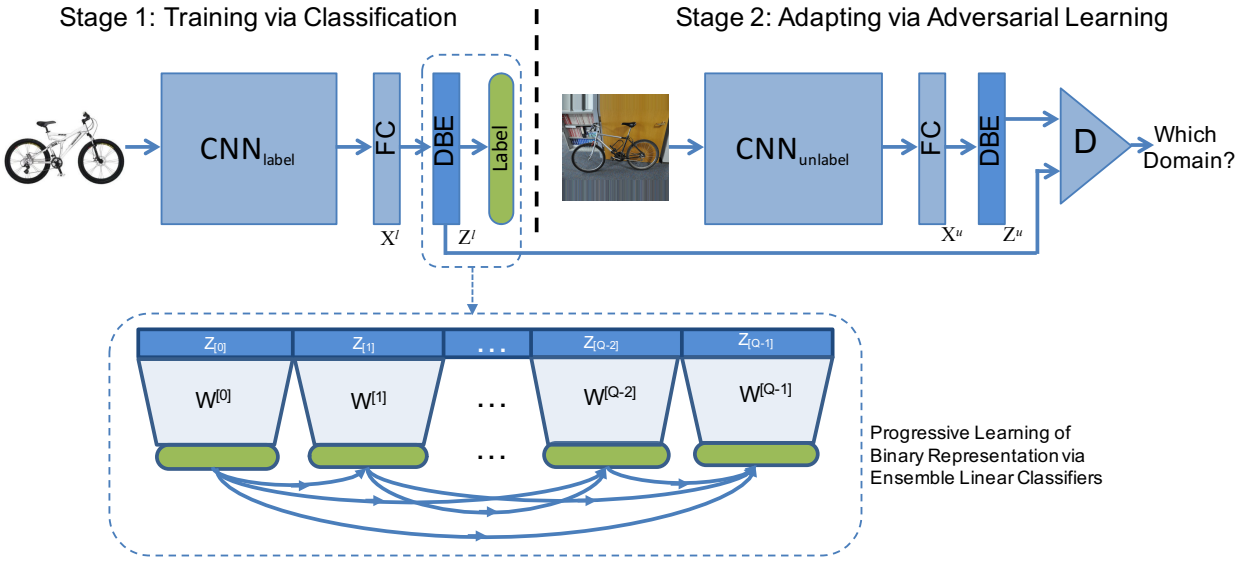


Figure 5.1: The overall architecture of the proposed p-ABR. Images of source (labeled) and target (unlabeled) domains are fed into two separate nonlinear projections serving as hashing function, including CNN, fully-connected layer (FC), Direct Binary Embedding layer (DBE). In addition, a discriminator (D) is employed to determine which domain the DBE layer activation belongs to. We use 2-stage training to obtain the hashing functions for two domains. In stage 1, network for the labeled domain (CNN together with FC layer and DBE layer) is trained via classification task; in stage 2, network for the unlabeled domain is updated via adversarial learning while fixing the labeled domain network. The progressive learning of binary representation is shown in the dashed box. The blue blocks are binary representations, the green blocks are the output of linear classifiers, i.e., predictions, the arrows indicate the direction along which the previous prediction is added to. Best viewed in color.

In the following, we elaborate on how to design the different loss components regarding progressive learning of discriminative binary representation (Sec. 5.4.1) and minimize the domain discrepancy via the domain adaptation (Sec. 5.4.2).

5.4.1 Progressive Learning of Various Code-length Discriminative Binary Representation

In order to learn discriminative binary representation effectively, we consider decomposing binary representation into individual blocks on which an ensemble of linear classifiers are deployed. Alternatively, $\mathbf{B}^l$ for the labeled domain can be expressed by column vectors $\mathbf{B}^l = [\mathbf{B}^{l,0} \dots \mathbf{B}^{l,j} \dots \mathbf{B}^{l,L-1}]$ where $\mathbf{B}^{l,j} \in \{0,1\}^{N\times 1}$ $(0 \leq j \leq L-1)$ is the ith bit column vector for $\mathbf{B}$[1]. For the purpose of learning Q different code-lengths binary representations indexed by q, we define

$$\begin{aligned} \mathbf{B}_q^l &= [\mathbf{B}^{l,0} \dots \mathbf{B}^{l,L(q+1)/Q-1}],\ 0 \leq q \leq Q-1 \\ &= [\mathbf{B}_{[0]}^l \dots \mathbf{B}_{[q]}^l] \end{aligned} \tag{5.2}$$

where $\mathbf{B}_{[q]}^l = [\mathbf{B}^{l,Lq/Q} \dots \mathbf{B}^{l,L(q+1)/Q-1}]$. This is illustrated in Fig. 5.2. For instance, when $L = 32$, $Q = 2$, we have $\mathbf{B}_0^l = [\mathbf{B}^{l,0} \dots \mathbf{B}^{l,15}]$, $\mathbf{B}_1^l = [\mathbf{B}^{l,0} \dots \mathbf{B}^{l,31}]$, $\mathbf{B}_{[0]}^l = [\mathbf{B}^{l,0} \dots \mathbf{B}^{l,15}]$, $\mathbf{B}_{[1]}^l = [\mathbf{B}^{l,16} \dots \mathbf{B}^{l,31}]$.

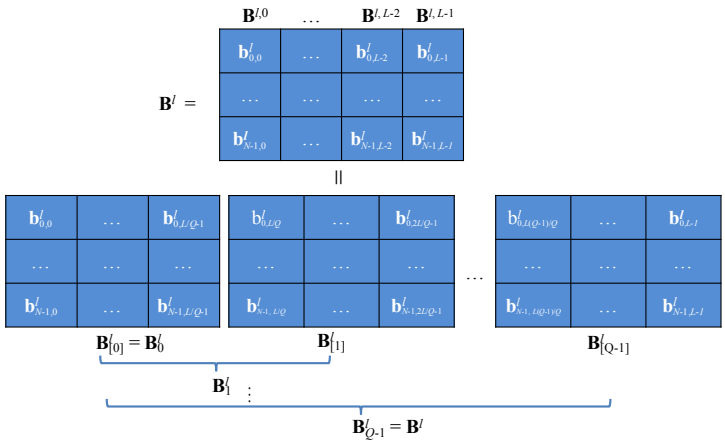


Figure 5.2: Divide the representation into column-wise blocks.

[1]We denote $\mathbf{B}^0$ as the most significant bit

To learn such binary representations, denote the ensemble of Q linear classifiers with weight $\mathbf{W} = [\mathbf{w}^{[0]}; \ldots; \mathbf{w}^{[Q-2]}; \mathbf{w}^{[Q-1]}] \in \mathbb{R}^{L \times C}$, where $\{\mathbf{w}^{[q]}\}_{q=0}^{Q-1} \in \mathbb{R}^{L/Q \times C}$ is a linear classifier for $\mathbf{B}_{[q]}^{l}$. C is the number of categories. Naturally, the prediction posterior can be used as softmax of the output from the linear classifiers, i.e., $p(\mathbf{Y}^l|\mathbf{B}^l) = \exp(\mathbf{B}^l\mathbf{W}_{\mathbf{Y}^l}) / \sum_{j=1}^{C} \exp(\mathbf{B}^l\mathbf{W}_j)$. We assume that $\{\mathbf{B}_{[q]}\}_{q=0}^{Q-1}$ are independent. Consider the prior $p(\mathbf{Y}^l)$ is fixed and we treat each individual binary representation block equally, the posterior for $\mathbf{B}^l$ can be expressed using Bayes' theorem,

$$
\begin{aligned}
p(\mathbf{Y}^l|\mathbf{B}^l) &\propto p(\mathbf{Y}^l) \prod_{q=0}^{Q-1} p(\mathbf{B}_{[q]}^l|\mathbf{Y}^l) \\
&= \frac{1}{p(\mathbf{Y}^l)^{Q-1}} \prod_{q=0}^{Q-1} p(\mathbf{Y}^l|\mathbf{B}_{[q]}^l) p(\mathbf{B}_{[q]}^l) \\
&\propto \prod_{q=0}^{Q-1} p(\mathbf{Y}^l|\mathbf{B}_{[q]}^l) \\
&\propto \prod_{q=0}^{Q-1} \exp\left(\mathbf{B}_{[q]}^l \mathbf{w}_{\mathbf{Y}^l}^{[q]}\right) \\
&= \exp\left(\sum_{q=0}^{Q-1} \mathbf{B}_{[q]}^l \mathbf{w}_{\mathbf{Y}^l}^{[q]}\right)
\end{aligned}
\tag{5.3}
$$

Furthermore, in order to make $p(\mathbf{Y}^l|\mathbf{B}_{[q]}^l)$ a stronger evidence for the ensembled final prediction as well as learning binary representation of various code-lengths simultaneously, we also learn shorter binary representation separately. Therefore, we end up with a progressive learning strategy by iteratively using Eq. 5.3 as posterior made by representations of all possible code lengths:

$$
\begin{aligned}
\mathcal{L}_{\text{supervised}} &= \sum_{q=0}^{Q-1} \mathcal{L}_q = \sum_{q=0}^{Q-1} \sum_{i=0}^{N-1} y_i^l \log p(y_i^l|\mathbf{b}_{i,q}^l) \\
s.t.\ p(y_i^l|\mathbf{b}_{i,q}^l) &= \frac{\exp\left(\sum_{k=0}^{q} \mathbf{b}_{i,[k]}^l \mathbf{w}_{y_i^l}^{[k]}\right)}{\sum_{j=1}^{C} \exp\left(\sum_{k=0}^{q} \mathbf{b}_{i,[k]}^l \mathbf{w}_j^{[k]}\right)}
\end{aligned}
\tag{5.4}
$$

Considering the discrete nature of $\mathbf{B} = threshold(\cdot)$, directly minimizing $\mathcal{L}$ is difficult. We adopt a Direct Binary Embedding (DBE) [53] layer after the final fully-connected layer

of CNN to learn a binary-like representation, thus achieve effective continuous relaxation on the discrete sgn(·) function without quantization loss counting for the error between the binary code and its relaxation. Denote the DBE layer as f_{DBE}, the output of the finally fully-connected layer and DBE layer as $\mathbf{X}$ and $\mathbf{Z}$, respectively, we have

$$\begin{aligned}\mathbf{Z} &= H(I,\theta) = f_{\text{DBE}}(\mathbf{X}) \\ &= \tanh(\text{ReLU}(\text{BN}(\mathbf{X}\mathbf{W}_{\text{DBE}} + b_{\text{DBE}})))\end{aligned} \tag{5.5}$$

where $\mathbf{W}_{\text{DBE}}$ maps the latent CNN features $\mathbf{X}$ into L-dimensional space and b_{DBE} is the bias; BN represents batch normalization. Similar to the decomposition of $\mathbf{B}^l$, we have $\mathbf{Z}^l = [\mathbf{Z}^{l,0} \dots \mathbf{Z}^{l,L(q+1)/Q-1}]$, $0 \le q \le Q-1$, and we choose to minimize the following supervised loss during training:

$$\begin{aligned}\mathcal{L}_{\text{supervised}} &= \sum_{q=0}^{Q-1}\sum_{i=0}^{N-1} y_i^l \log p(y_i^l|\mathbf{z}_{i,q}^l) \\ s.t.\ p(y_i^l|\mathbf{z}_{i,q}^l) &= \frac{\exp\sum_{k=0}^{q}\mathbf{z}_{i,[k]}^l \mathbf{w}_{y_i^l}^{[k]}}{\sum_{j=1}^{C}\exp\sum_{k=0}^{q}\mathbf{z}_{i,[k]}^l \mathbf{w}_j^{[k]}}\end{aligned} \tag{5.6}$$

Extending to Multilabel Datasets By treating each label independently during the classification task, logistic regression is commonly used for multilabel image classification. Eq. 5.3 can naturally used for multilabel datasets scenario:

$$\begin{aligned}\mathcal{L}_{\text{supervised}}^{\text{multilabel}} = &\sum_{q=0}^{Q-1}\sum_{i=0}^{N-1}\sum_{p=1}^{C}\big(y_{i,p}^l \log p(y_{i,p}^l|\mathbf{z}_{i,q}^l) \\ &+(1-y_{i,p}^l)\log(1-p(y_{i,p}^l|\mathbf{z}_{i,q}^l))\big) \\ s.t.\ p(y_{i,p}^l|\mathbf{z}_q^l) = &\frac{1}{1+\exp\sum_{k=0}^{q}\mathbf{z}_{i,[k]}^l \mathbf{w}_p^{[k]}}\end{aligned} \tag{5.7}$$

5.4.2 Domain Adaptation for Binary Representation

$\mathcal{L}_{\text{unsupervised}}$ aims to reduce the distributional discrepancy between binary representations of different domains via adversarial domain adpation. Among domain adpatation methods, adversarial learning reinforces the distribution of the unlabeled domain feature $\mathbf{Z}^u$ to match

that of the labeled domain feature $\mathbf{Z}^l$ exactly [23]. In contrast, we argue that metric approaches such as MMD and CMD are not expressive enough because they optimize the statistics (e.g., moments) rather than domain distribution itself. For example, MMD minimizes the difference of distribution measure expectations [24]; CMD explicitly matches the moments for each order [89]. Consequently, we choose to let discriminator determine which domain it is based on the DBE-layer output directly. Adversarial learning leads to better hashing function H^u for the unlabeled domain $\mathcal{D}_u$ as it learns a better DBE-layer for $\mathcal{D}_u$. Formally, we choose to optimize the following problem for unsupervised domain adaption

$$\begin{aligned} \min \mathcal{L}^{(\text{adv})}_{\text{unsupervised}} := \min_{H^u} \max_{D} & \mathbb{E}_{\mathbf{Z}^l \sim \mathcal{D}_l}[\log D(\mathbf{Z}^l)] \\ & + \mathbb{E}_{\mathbf{Z}^u \sim \mathcal{D}_u}[\log(1 - D(\mathbf{Z}^u))] \\ \text{s.t. } \mathbf{Z}^l = & H(I^l, \theta^l),\ \mathbf{Z}^u = H(I^u, \theta^u) \end{aligned} \tag{5.8}$$

Equivalently, H^u and D can be optimized by:

$$\min_{D} -\mathbb{E}_{\mathbf{Z}^l \sim \mathcal{D}_l}[\log D(\mathbf{Z}^l)] - \mathbb{E}_{\mathbf{Z}^u \sim \mathcal{D}_u}[\log(1 - D(\mathbf{Z}^u)] \tag{5.9}$$

$$\min_{H^u} -\mathbb{E}_{\mathbf{Z}^u \sim \mathcal{D}_u}[\log D(\mathbf{Z}^u)] \tag{5.10}$$

Note that we choose to optimize $\log D(\mathbf{Z}^u)$ for H^u since it provides larger gradient.

Alternative Domain Adaptation Methods Meanwhile we choose to compare with two popular domain adaptation methods: **MMD** and **CMD** as the domain adaptation methods. They are defined as:

$$\mathcal{L}^{(\text{MMD})}_{\text{unsupervised}} := \min_{\Theta^{p(q)}} \frac{1}{N^2} \sum_{i,j=1}^{N} k(\mathbf{z}_i^p, \mathbf{z}_j^p) + \frac{1}{M^2} \sum_{i,j=1}^{M} k(\mathbf{z}_i^q, \mathbf{z}_j^q) - \frac{2}{NM} \sum_{i,j=1}^{N,M} k(\mathbf{z}_i^p, \mathbf{z}_j^q) \tag{5.11}$$

$$\begin{aligned} \mathcal{L}^{(\text{CMD})}_{\text{unsupervised}} &:= \min_{\Theta^{p(q)}} \frac{1}{|b-a|} \|\mathbf{E}(\mathbf{Z}^p) - \mathbf{E}(\mathbf{Z}^q)\|_2 + \sum_{k=2}^{K} \frac{1}{|b-a|^k} \|C_k(\mathbf{Z}^p) - C_k(\mathbf{Z}^q)\|_2 \\ &= \|\mathbf{E}(\mathbf{Z}^p) - \mathbf{E}(\mathbf{Z}^q)\|_2 + \sum_{k=2}^{K} \|C_k(\mathbf{Z}^p) - C_k(\mathbf{Z}^q)\|_2 \end{aligned} \tag{5.12}$$

where $k(\cdot, \cdot)$ is the Gaussian kernel function with bandwidth γ, i.e., $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$; $C_k(\mathbf{Z}) = \mathbf{E}((\mathbf{z} - \mathbf{E}(\mathbf{Z}))^k)$ is the kth central moment; a and b are the lower and upper

bound of the DBE activation $\mathbf{Z}^{p(q)}$, and $0 \leq |\mathbf{Z}| < 1$. Naturally, we set $a = 0$ and $b = 1$, i.e., $|b - a| = 1$. K is the number of moments considered in the model.

5.5 Experiments

To evaluate p-ABR, we conduct extensive experiments on various tasks including unlabeled domain prediction/classification and cross-domain image retrieval, as well as open set domain adaptation tasks. For prediction task, we compare against several state-of-the-art unsupervised algorithms (floating number features), including CMD [89], Deep CORAL [79], DAN [57] and RTN [58], as well as hashing algorithms with different domain adaptation methods, including DAH [81] and MMD based and CMD based progressive binary representation learning algorithms, denoted as p-MBR and p-CBR, respectively; for image retrieval task, we compare with state-of-the-art domain adaptive hashing algorithms and hashing with different domain adaptation methods (DAH, p-MBR, p-CBR).

The datasets explored in the experiments are the Office dataset [73] for object recognition, SVHN and MNIST datasets for digit recognition, and finally MIRFLICKR [30] and MS COCO [48] for multilabel image prediction/retrieval under the open set domain adaptation settings.

We implement p-ABR and several compared methods in PyTorch[2]. ResNet-18 [29] is the main CNN model used in p-ABR and several compared methods. The discriminator has three fully connected layer with dimension of 256, 128, 64, each of which has batch normalization layer before ReLU. During training, we use pretrained CNN weights on ImageNet as initialization. For the Office dataset, we train the network on the labeled domain for 1,000 iterations with batch size of 32 during stage 1; then we train the unlabeled network for 100 iterations for adaptation. The batch size is 32. We use the learning rate of 5e-5 for CNN, 1e-6 for the discriminator network, and 5e-4 for the linear classifiers. The reason of choosing small learning rate is to prevent potential instability caused by discriminator and to not to saturate the DBE layer. For SVHN and MNIST dataset, with the ImageNet pretrained weights we train the network for 2,000 iterations with batch size of 64 in stage 1,

[2]https://github.com/pytorch/pytorch

Table 5.1: Comparison of unsupervised cross-domain retrieval performance in terms of mAP score on the Office dataset over various code lengths. Models with both shared weights and unshared weights are compared. The best accuracy is shown in boldface, and the second best is underlined.

Weights	Method	A→W			A→D			D→W		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
shared	ResNet-B	0.417	0.421	0.541	0.416	0.459	0.513	0.793	0.823	0.897
	DAH [81]	0.521	0.593	0.647	<u>0.534</u>	<u>0.622</u>	<u>0.680</u>	0.821	0.891	0.927
	p-CBR	0.467	0.502	0.521	0.420	0.498	0.532	<u>0.844</u>	<u>0.901</u>	<u>0.938</u>
	p-MBR	0.454	0.521	0.560	0.405	0.501	0.552	0.821	0.868	0.918
	p-ABR	0.486	0.547	0.583	0.448	0.487	0.528	0.828	0.880	0.922
Unshared	p-CBR	0.513	0.528	0.547	0.445	0.498	0.539	**0.886**	**0.923**	**0.944**
	p-MBR	<u>0.531</u>	<u>0.603</u>	<u>0.655</u>	0.529	0.617	0.669	0.769	0.838	0.902
	p-ABR	**0.533**	**0.628**	**0.689**	**0.538**	**0.631**	**0.685**	0.798	0.883	0.929
		W→A			D→A			W→D		
shared	ResNet-B	0.374	0.418	0.451	0.371	0.441	0.502	0.892	0.916	0.930
	DAH [81]	0.462	0.547	0.573	0.491	0.523	0.571	<u>0.925</u>	**0.961**	**0.990**
	p-CBR	0.412	0.471	0.515	0.403	0.460	0.493	**0.926**	0.951	0.973
	p-MBR	0.404	0.513	0.579	0.448	0.502	0.566	0.901	<u>0.955</u>	<u>0.981</u>
	p-ABR	0.411	0.470	0.505	0.434	0.478	0.483	0.864	0.898	0.926
Unshared	p-CBR	0.443	0.505	0.547	0.437	0.501	0.539	0.921	0.937	0.944
	p-MBR	<u>0.475</u>	<u>0.563</u>	<u>0.607</u>	<u>0.501</u>	<u>0.559</u>	<u>0.591</u>	0.907	0.928	0.948
	p-ABR	**0.504**	**0.572**	**0.619**	**0.523**	**0.597**	**0.636**	0.887	0.931	0.967

and 1,200 iterations for adaptation in stage 2. For the MS COCO dataset and MIRFLICKR dataset, similarly we train the network for 2,000 iterations in stage 1 and 1,000 in stage 2. For compared methods, we set $\lambda = 0.3$ and $\gamma = 0.5$ for p-MBR as recommended in RTN [58]; we set $\lambda = 1$ and $K = 5$ for p-CBR as recommended in CMD [89]; We also reimplement ADDA and DAH using ResNet-18 as CNNs.

5.5.1 Office Dataset

The **Office** dataset [73] consists of color images associated with 31 labels from three distinct domains: amazon (A), webcam (W) and dslr (D). It is a *de facto* standard for domain adaptation algorithms. Following previous works, we consider domain shifts between all domain pairs. Additionally, we also compare with domain adaptation hashing algorithms with shared CNN to show the benefit of using separate CNNs for different domains.

We report the performance comparison of cross-domain retrieval tasks using mean average precision (mAP). From mAP scores reported in Table 5.1, it is obvious that using domain adaptation unanimously improves upon direct transfer learning, i.e., ResNet-B. Furthermore,

p-ABR without weight sharing generally outperforms other methods across various code lengths, especially for two domains with relatively large domain shift (A→W, A→D, W→A, D→A[3]). When two domains are very similar (D→W, W→D), DAH, p-MBR and p-CBR outperform p-ABR by a small margin. We argue that it is related to instability during training caused by the discriminator. And such instability becomes more dominant if two domains are very simila. Recent studies of generative adversarial networks (GAN) also observe similar instability during adversarial learning [3, 4].

Meanwhile, it can also be observed that methods without sharing weights provide superior performance. This is because using separate CNNs for individual domains captures better domain dependent information at lower level of the networks. However approaches with shared weights provide competitive or even superior performance for similar domains (D→W, W→D), confirming the visual similarity between them.

We also evaluate the proposed algorithm on the task of prediction in the unlabeled domain. The result is summarized in Table 5.2. The proposed p-ABR achieves competitive average prediction accuracy with 64-bit binary representations, outperforming most state-of-the-art domain adaptation methods that are based on floating number features, including RTN, DAN and dCORAL. And CMD and ADDA only outperform p-ABR marginally. Note that CMD [89] originally uses VGG16 model [2], which provides higher performance than ResNet-18 that is used in p-ABR[4]. Finally, we can see that p-ABR provides the best performance among all compared hashing/binary representation methods.

5.5.2 SVHN and MNIST Datasets

SVHN and MNIST datasets are image datasets containing digits (0 to 9) from street view and handwriting, respectively. We use the full training sets of both datasets in the experiment and we consider adaptation of SVHN→MNIST and MNIST→SVHN. Furthermore, in addition to using resnet-18 as CNN in the method, we also use LeNet, which is commonly used in digits domain adaptation, as an alternative to provide fair comparison.

[3]Following the notation of domain adaptation, we use X→Y to denote using images from labeled domain X to retrieve images from unlabeled domain Y

[4]https://github.com/jcjohnson/cnn-benchmarks

Table 5.2: Comparison of prediction accuracy on the unlabeled domain on the Office dataset. For the hashing methods, code length of 64 bits is used; shared weights (s) are also included in the comparison.

Method	A→W	D→W	W→D	A→D	D→A	W→A	average
dCORAL [79]	0.664	0.957	0.992	0.668	0.528	0.515	0.721
DAN [57]	0.685	0.960	0.990	0.670	0.540	0.531	0.729
RTN [58]	0.733	**0.968**	**0.996**	0.710	0.505	0.510	0.737
CMD [89]	**0.770**	0.963	0.992	**0.796**	0.638	0.633	0.799
ADDA [80]	0.753	0.962	0.990	0.734	**0.701**	**0.663**	**0.801**
DAH [81]	0.681	0.941	0.992	0.668	0.558	0.521	0.726
p-CBR (s)	0.521	0.945	0.978	0.581	0.470	0.496	0.665
p-MBR (s)	0.569	0.924	0.979	0.574	0.554	0.579	0.696
p-ABR (s)	0.682	0.943	0.957	0.587	0.497	0.501	0.695
p-CBR	0.552	0.941	0.984	0.574	0.479	0.516	0.674
p-MBR	0.657	0.923	0.949	0.655	0.575	0.571	0.721
p-ABR	0.695	0.917	0.969	0.782	0.659	0.634	0.776

Table 5.3 shows the results on prediction in the unlabeled domain for both adaptation directions and the code-length is 64 bits for binary representation learning/hashing algorithms. p-ABR achieves comparable state-of-the-art results as ADDA. Even using LeNet as CNN, p-ABR(L) still achieves better results than most compared methods. Not only does this result confirm the advantage of adversarial domain adaptation over other metric-based adaptation, also it suggests that discriminative binary representation has the same capability as conventional floating number features.

Table 5.3: Comparison of prediction on adapted SVHN and MNIST, both adaptation directions are included. The results are based on code-length of 64 bits.

	DRCN [17]	DAN [57]	ADDA [80]	p-CBR	p-MBR	DAH	p-ABR(L)	p-ABR
SVHN→MNIST	0.820	0.739	**0.881**	0.673	0.734	0.783	0.791	0.873
MNIST→SVHN	0.401	-	0.453	0.322	0.381	0.391	0.412	**0.482**

We also conduct experiments on cross-domain retrieval on SVHN and MNIST datasets. The experiment is summarized in Table 5.4. Similar to the observations in Sec. 5.5.1, p-ABR provides superior performance in terms of mAP on retrieval tasks across various code-lengths.

Table 5.4: Comparison of unsupervised cross-domain retrieval performance in terms of mAP score on the SVHN and MNIST dataset over various code lengths. Models with both shared weights and unshared weights are compared.

Methods	SVHN→MNIST				MNIST→SVHN			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
ResNet-B	0.572	0.619	0.632	0.639	0.325	0.388	0.394	0.402
DAH [81]	0.719	0.801	0.816	0.820	0.466	0.483	0.502	0.503
p-CBR	0.698	0.740	0.756	0.761	0.430	0.451	0.472	0.478
p-MBR	0.726	0.792	0.821	0.822	0.471	0.498	0.511	0.515
p-ABR	**0.731**	**0.823**	**0.840**	**0.846**	**0.499**	**0.509**	**0.536**	**0.532**

Table 5.5: Retrieval performance (mAP) on MIRFLICKR retrieving both MS COCO training set (10,000 samples) and validation set.

Methods	MIRFLICKR → COCO-tr			MIRFLICKR → COCO-val		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
ResNet-B	0.353	0.356	0.354	0.342	0.348	0.351
p-CBR	0.417	0.398	0.443	0.397	0.419	0.438
p-MBR	0.458	0.471	0.485	0.450	0.469	0.487
p-ABR	**0.460**	**0.485**	**0.502**	**0.458**	**0.486**	**0.498**

Table 5.6: Unsupervised image retrieval on the MS COCO dataset over various code lengths.

length	ITQ [21]	DeepBit [46]	ResNet-B	p-CBR	p-MBR	p-ABR
16 bits	0.364	0.411	0.405	0.415	0.442	**0.453**
32 bits	0.365	0.412	0.407	0.418	0.441	**0.455**
64 bits	0.368	0.410	0.409	0.421	0.441	**0.456**

5.5.3 MIRFLICKR and MS COCO Datasets

While unsupervised domain adaptation focuses on different domains of images having identical semantics, in real-world applications, it is more probable that images from different domains are only partially overlapped semantically, i.e., the open set domain adaptation problem. By experiments we show that p-ABR can improve cross-domain retrieval in such scenario. Note that the setting of open domain adaptation in this experiment is different from previous work such as ATI [65]. We assume that the knowledge of the labeled domain is fully gained and we implicitly focus on domain adaptation caused by visual variety rather than semantics. The MIRFLICKR and MS COCO datasets are used in this experiment. We follow experimental settings of previous hashing methods [10, 35], 5,000 images are randomly sampled from the MIRFLICKR training set to form the labeled domain and 10,000 images

are sampled from the training set of MS COCO to form the unlabeled domain; furthermore, the validation set of MS COCO is included in the experiment as well.

In order to evaluate the relevance of the retrieved images from the unlabeled domain, it is necessary to merge their labeling. We treat MIRFLICKR dataset as the labeled domain and use their labels as reference and convert the labeling of MS COCO into similar format for evaluation purpose during testing. Specifically, we augment MIRFLICKR by one extra label accounting for the semantic information that is in MS COCO but not in MIRFLICKR, and vice versa; then we manually map MS COCO annotations into MIRFLICKR label space according to their semantic similarity. For instance, "Apple" and "Sandwich" from MS COCO are considered "Food" in MIRFLICKR; "Spoon" and "Knife" do not have any similar concept in MIRFLICKR, and they are considered as the augmented label. For more details of label merging, please refer to the supplementary material.

We compare the performance of p-ABR, p-MBR and p-CBR on the task of retrieving images from partially relevant unlabeled domain in terms of mAP, and the comparison is shown in Table 5.5. Since both MIRFLICKR and MS COCO are multilabel datasets, we consider a retrieval successful as long as one label matches. As suggested by empirical evidence in Section 5.5.1, only methods with unshared weights are include in the comparison as the shared weights counterparts do not perform as well in the scenario where domain shift is large. Table 5.5 reports the retrieval performance comparison based on mAP. Clearly p-ABR achieves the best retrieval result across various code lengths. Specifically p-ABR gains 30% of retrieval performance at the code length of 16 bits, and over 40% at 64 bits; the retrieval is also effective when retrieving relevant images from the unseen images in the validation set of MS COCO dataset. This indicates that p-ABR provides consistent performance as long as retrieving images from the same unlabeled domain.

We also conduct experiments to show that p-ABR can improve unsupervised image retrieval after the adaptation given a model pretrained on the dataset of a labeled domain. MIRFLICKR and MS COCO datasets are considered the labeled and unlabeled datasets in the experiment. We choose PCA-ITQ [21] as non-end-to-end baselines. ResNet-18 (trained on MIRFLICKR) features are extract as input for LSH and PCA-ITQ. Meanwhile we also compare with an end-to-end unsupervised binary descriptor DeepBit [46], which is

implemented using ResNet-18. 10,000 samples are randomly selected from the training set of MS COCO to form the database and the entire validation set is used as the query set. Table 5.6 reports the mAP score of the unsupervised retrieval task. It can be concluded that p-ABR achieves the best retrieval performance. DeepBit achieves similar performance as ResNet-B without domain adaptation; p-ABR improves ResNet-B by 5%. This is because the domain adaptation aligns the shared semantics of the two domains, yielding an effective unsupervised uni-domain retrieval method.

5.5.4 Ablation Study

In this section, we study the benefit of progressive learning. Unsupervised domain adaptation is inherently complex, and learning high-quality features for the labeled domain facilitates the performance of adapted unlabeled domain significantly. By experimental evidence, we show that progressive learning improves the representation quality in domain adaptation.

As mentioned before, progressive learning decomposes the discriminative binary representation learning into smaller subtasks (shorter representation learning), each of which is simultaneously used as strong guidance for a larger and more complex subtasks (longer representation learning); meanwhile, the larger subtasks distill the learned information to the smaller subtasks. To validate this, we consider prediction tasks in both SVHN-MNIST datasets and Office datasets. Specifically, we compare progressive learning and conventional learning of binary representation, i.e., without progressively using prediction made by shorter representation (denote as Adversarial Binary Representation, ABR), on amazon→dslr and SVHN→MNIST adaptation directions. Visualized in Fig. 5.3, the comparison shows that progressive learning achieves higher prediction accuracy in the unlabeled domain consistently. Furthermore, the progressive learning improves the shorter representation performance greatly.

5.6 Conclusion

In this work, we studied the problem of unsupervised cross-domain image hashing and propose Adversarial-Bit, which effectively learns binary representations for labeled and

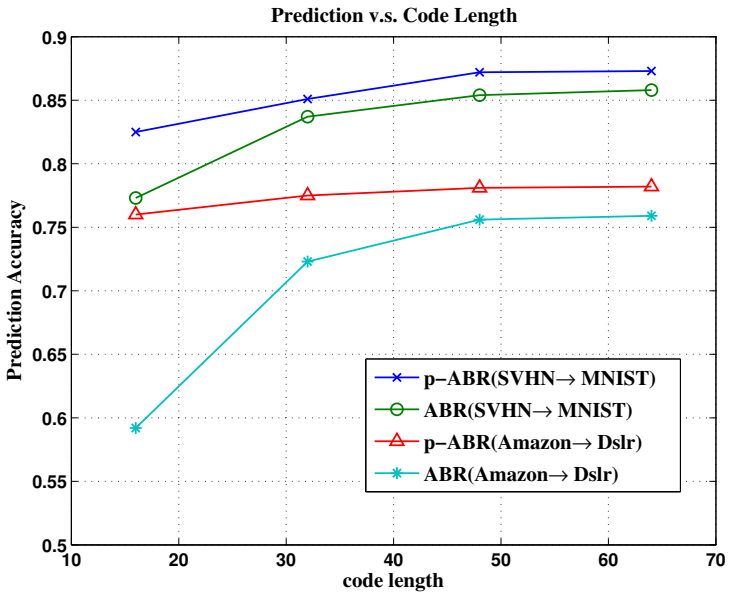


Figure 5.3: Comparison between conventional binary representation and progressive binary representation for domain adaptation.

unlabeled domain via adversarial domain adaptation. We compare with other approaches such as MMD and CMD, as well as weight sharing of models between domains. Extensive experiments validated the superiority of the proposed Adversarial-Bit in several tasks including unsupervised cross-domain image retrieval, cross-domain prediction and unsupervised image retrieval.

Chapter 6

Instance-level Binary Representation Learning

6.1 Abstract

In this work, we study the problem of instance-level binary representation learning. Most recent works of image hashing and binary representation learning focus on the global representation for images. While this is effective for fast and accurate image retrieval and understanding, it does not provide a finer granularity insight of images, i.e., the instances that actually contribute to the semantics of the images. As a result, we propose Det-Bit, a novel approach of learning binary representation for instances. We build our approach on top of Faster RCNN, a state-of-the-art object detection architecture. By separating the streams for bounding box regression and object classification, only the semantic information is preserved in the binary representation. Extensive experiments validates the performance of Det-Bit on several tasks, including object detection, multi-instance retrieval.

6.2 Introduction

Learning binary representation for images has been extensively studied, due to its computational efficiency and high performance on tasks such as classification, retrieval, etc. Powered by deep neural networks, recent binary representation methods (often referred to as image hashing) learn *global representations* for images based on similarity or discriminative information. (some examples). Albeit the success on recognition or retrieval of images, these methods are less effective on finer-granularity tasks, e.g., instance recognition/retrieval, object localization.

In order to perform instance-level tasks, we consider learning binary representations under the framework of object detection. Object detection has been studied for long and many effective end-to-end algorithms have been proposed to embrace the strong expressiveness of deep neural network. These methods can be categorized into two main directions: 1-stage detectors and 2-stage detectors. Usually 2-stage detectors achieves higher performance by adopting an extra stage of object proposal, which is usually fulfilled by a Regional Proposal Network (RPN). Based on the proposed regions, Region of Interest (RoI) pooling is performed on the deep features to generate instance-level features, and are used to regress the bounding box (BBox) and classify the specific object (classifier). Such extracted regional features is

very helpful to generate binary representations for potential object. (Reason of choosing this over 1-stage methods).

In this work, we integrate the binary representation learning into the object detection framework, such that it is possible to perform object detection, instance recognition/retrieval and image-level retrieval at the same time. (benefit of focusing on instance). Although naturally CNNs are capable of focusing on objects due to its class-wise training (cite, MIT 2016 cvpr paper), using instance-level information reduces the influence of unrelated background and noise (cite). We show that the binary representation learned by the proposed method provides competitive performance on object detection, i.e., recognizing and localizing objects.

6.3 Related Works

Object Detection: Early works on object detection, usually falling into sliding-window paradigm, rely on handcrafted features. They are quickly superseded by later CNN-based detectors with the rise of deep learning. These detectors can be categorized as one-stage and two-stage approaches. Inheriting from sliding-window based detectors, **One-stage** detectors, such as YOLO v2 [70], SSD [54], use a single CNN as a regressor and classifier with combination of using anchor boxes with different scales and ratio. Since they output the bounding boxes and the category of the objects directly without any intermediate stage, one-stage detectors are fast, but often fall behind on the performance comparing to two-stage counterparts. Regional CNN detectors, as typical examples of **Two-stage** detectors, are based on proposing potential object regions. R-CNN uses Selective Search as a separate step for object proposal purpose. Fast RCNN improves R-CNN by sharing a single CNN for all object proposals. Faster RCNN further proposed a Regional Proposal Network (RPN) to integrate the object proposing into CNN, improving the detecting speed and accuracy. For the purpose of instance-level image understanding, the features for potential instances within images are free to obtain from two-stage detectors. This enables us to learn binary representations for objects efficiently.

Image Hashing: Image hashing has been well studied. It aims to learn compact binary codes for image sets to achieve fast and accurate retrieval task. From the early works of using handcrafted features as input () to more recent end-to-end methods, image hashing usually use similarity information to learn the codes. Common criteria include pairwise-similarity, triplet. Different quantization techniques are proposed as well. A quantization loss is often included for the relaxation of the discrete hashing function. Meanwhile, there are works focusing on learning good hashing function directly from continuous features of images. HashNet [10] introduces a scaling factor into the squashing nonlinearity for binary-like embedding learning, and gradually increases it to approximate the sign function when it is large enough; DBE [53] aims to learn 0-1 embedding directly by proposing a novel binary embedding layer and achieves great performance in retrieval and classification tasks. However, most the works focuses on the global binary representations for images and cannot provide information on a finer granularity. Recent works started to explore instance-level representation learning. For instance, DRH [77] generates regional features first, it converts them into binary codes and perform instance-level search and comparison together with query expansion.

Recently there are several works focusing on instance-level feature based hashing methods. DMIH [91] adopts the framework of SSD [54]. By considering learning hash code for regions and using multiple instance learning (MIL) [72], it learns a hash bag to index an image. DRH [76] learns regional and global hash codes under the framework of sliding window and regional proposal detectors. However, their experiments are only limited in building recognition datasets. Most previous works extract regional features and learn hash codes without considering object detection task, and their regional hash codes are usually redundant since there is no step such as non-maximum suppression (NMS). We aim to learn a hash code for each detected object, which is more compact and useful for detection purpose.

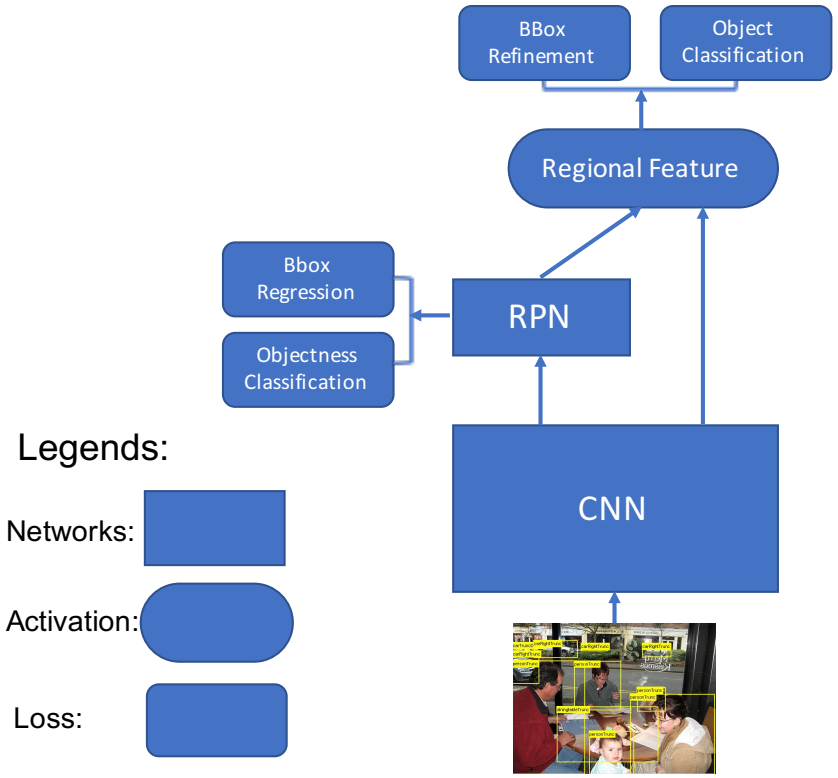


Figure 6.1: Overview of Faster R-CNN

6.4 Proposed Method

6.4.1 Revisit Faster RCNN

Faster R-CNN as the state-of-the-art object detector employs an effective regional proposing mechanism, i.e., RPN, for detecting potential objects in images. Based on the regional proposals, the object classification and BBox regression are performed. The architecture is shown in Fig. 6.1. By converting the ground truth BBox to the convolutional feature map size, making it computational efficient; meanwhile, by using the recently proposed RoIAlign layer [28] that faithfully preserves spatial locations of instances, better-quality instance features are extracted.

Since there are two stages in the detection process for Faster R-CNN, it inolves the optimization of several sub-problems. Although the original work [71] used an alternating training strategy, more recent experiment [86] discovers that it is possible to train the whole network altogether.

6.4.2 Supervised Learning of Binary Representation for Instances

For the well-labeled dataset (including both the database and potential queries), we propose to learn binary representations for instances in a supervised way. A natural way of learning binary representation for objects are binarizing the regional features directly. We achieve this by introducing a binarization layer between the RoI feature and the classifier (and the regressor). For the purpose of generating features for individual instances, we propose to separate the spatial information, i.e., the BBox and the semantics of instances by using two independent binary embedding layers for them, although the spatial information is not necessarily encoded in the binary codes. The binary codes for instances not only can be used for detection (localization and classification) purpose, they can also be used to retrieval images that contain relevant instances. The architecture of the proposed Det-Bit is shown in Fig. 6.2.

We argue that using the same set of codes for regression and classification is less effective when considering object retrieval task. Spatial information and semantics of the objects are encoded into separate binary codes, although they are pooled from the same RoIs. (Subject to experimental results).

6.4.3 Unsupervised Learning of Binary Representations for Instances

More often, we want to perform effective instance-level understanding for new datasets. Instead of using the features based on all the region proposals via RPN, we propose to focus on the regions with higher probability of objectness. Non-maximal suppression (NMS) is used to eliminate the regions with large overlap, so that one region is generated for each potential object/instance. Because there is no supervision to learn the binary representation as in previous section, we transfer the binary embedding layer, i.e., DBE layer, to the unsupervised learning.

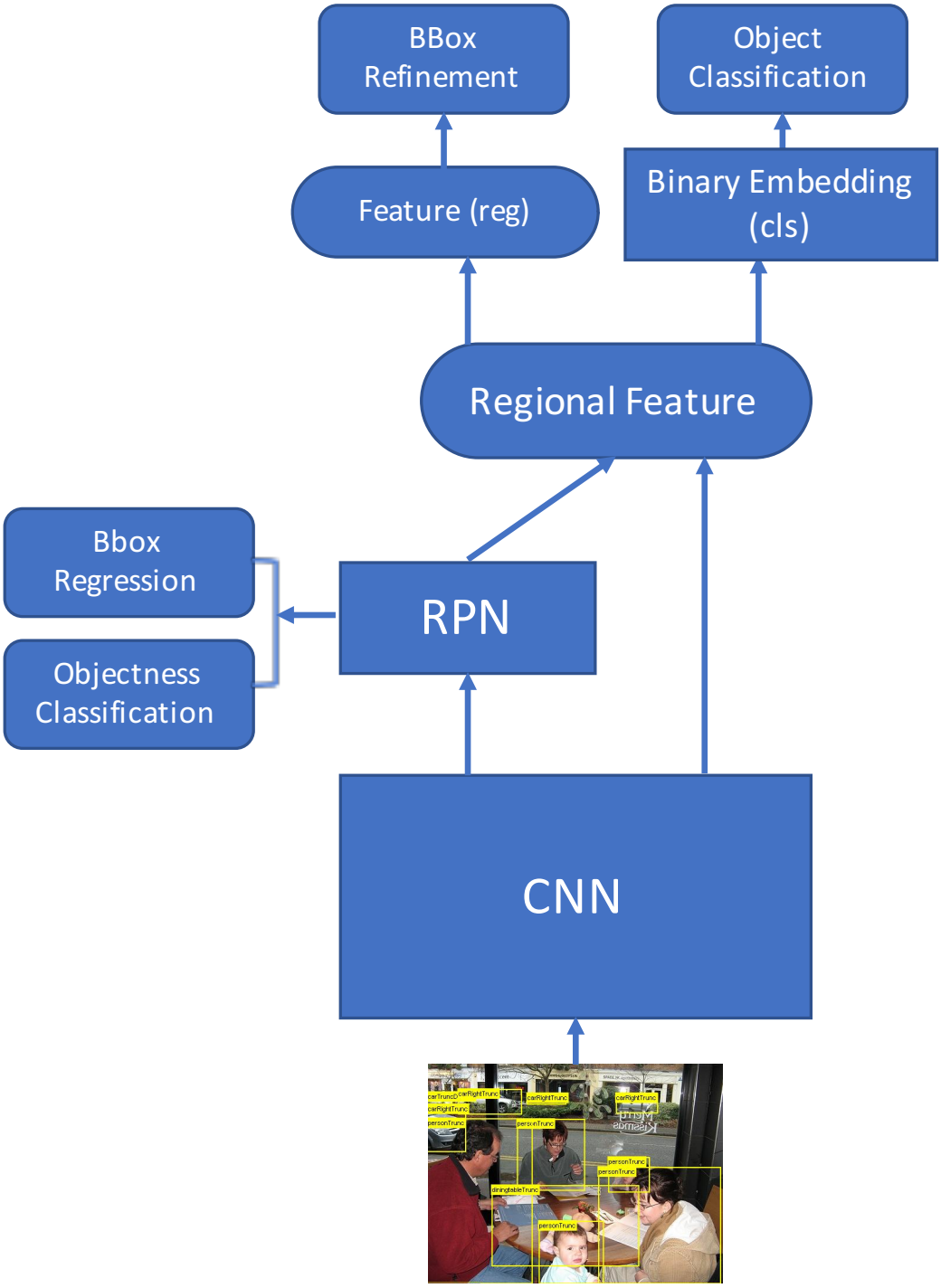


Figure 6.2: The overall architecture of the proposed Det-Bit

6.4.4 Learning Binary Representation

Previous works have well studied the problem of appropriate approximating the discrete $sgn(\cdot)$ function. Most them adopt a quantization loss and lose the hard thresholding directly. However, such loss introduces a bias to the original optimization problem. Meanwhile, some works consider a more sophisticated approach, aiming to learn binary code directly from continuous features without incorporating the quantization loss (cite). DBE learns binary representation by using an effective squashing nonlinearity to approximate the binary codes. HashNet introduces a gradual process to learn hashing function from continuity directly. In order to make DBE more effective, we propose DBE+ by embracing the gradual approximation, and it is defined as below.

$$\mathbf{Z} = f_{\text{DBE+}}(\mathbf{X}) = \tanh(\text{ReLU}(\text{BN}(\beta \cdot \mathbf{X}\mathbf{W}_{\text{DBE+}} + b_{\text{DBE+}}))) \tag{6.1}$$

6.5 Experiments

In this section, we evaluate the proposed Det-Bit. Our experiments are mainly based on PASCAL-VOC 2007 dataset [Everingham et al.].

PASCAL-VOC 2007 dataset has been used a benchmark dataset for object detection and multi-label image retrieval. Including both training and test set, there are around 10K color images containing objects of 20 categories. Not only do the images are of different sizes, they may also contain one or more instances in an image.

We compare Det-Bit with various state-of-the-art algorithms on different tasks to show its superiority. For object detection task, we mainly compare with the original Faster R-CNN to demonstrate that using binary code can achieve similar performance while reducing the space for object representation. For instance retrieval task, we compare with two multi-instance hashing algorithms, DMIH [91] and DSRH [90].

6.5.1 Experiments on Object Detection

In this section, we demonstrate that binary code can provide comparable performance on the object detection task comparing against original Faster R-CNN. We summarize the

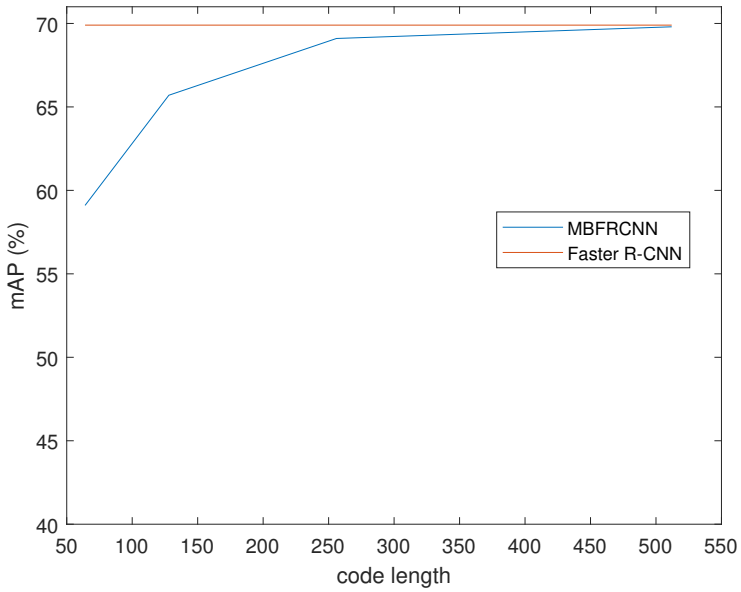


Figure 6.3: Comparison between Det-Bits and Faster R-CNN across code-length of 64, 128, 256, 512-bit

comparison in Fig. 6.3. It can be observed that when the code-length is 256-bit or longer, the performance gap is marginal. We can conclude that Det-Bit provides state-of-the-art detection performance while compressing space for instance representation. More detailed the detection results on each object category are in Table 6.1.

Table 6.1: Performance comparison of Faster R-CNN and MB-FRCNN (256-bit) on detection on PASCAL VOC 2007 test.

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair
FRCNN	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2
Det-Bit	69.1	69.6	78.8	65.9	55.1	50.0	75.8	82.4	79.9	46.7
cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
76.4	60.9	79.0	82.6	74.7	76.4	44.2	69.0	62.3	74.1	67.7

6.5.2 Experiments on Multi-Instance Retrieval

Instance-level binary representation not only is capable for object detection, it also enables retrieving images at finer granularity. Instead of using a global binary code to represent an image, a bag of codes is used instead. We process query images and database the same

way, i.e., using Det-Bit to obtain the bags of codes. We follow the experimental protocol in DMIH [91]. 1K query images are randomly sampled from the test set. Furthermore, at most 4 images are randomly picked from them to form the queries. Table 6.2 reports the comparison result against DMIH and DSRH. Note that we also include the result of single-instance retrieval. It is clear that Det-Bit provides a competitive performance on single object retrieval task, based on precision@100 metric. Meanwhile, Det-Bit improves the performance on the multi-instance retrieval task.

Table 6.2: Performance Comparison on Single object retrieval and multiple object retrieval (64-bit)

Metrics	Det-Bit	DMIH [91]	DSRH [90]
Precision@100	0.769	0.782	0.756
mAP (horse + person)	0.818	0.812	0.693
mAP (dog + cat)	0.803	0.799	0.642

6.6 Conclusions

In this work, we developed a novel binary representation learning method, namely Det-Bit, to learn binary codes for instances in images for the purpose of instance-level retrieval and object detection. By integrating the effective binary embedding layer DBE+ in to the state-of-the-art object detector Faster R-CNN, we split the sub-tasks of bounding box refinement and object classification, and use binary code only for the latter. As a result, Det-Bit learns semantic binary representation for the objects in the images. We evaluate Det-Bit with extensive experiments. For object detection task, Det-Bit provides comparable performance comparing with the original Faster R-CNN; for mutli-instance retrieval task, Det-Bit performs better than the state-of-the-art algorithms.

Chapter 7

Conclusion

Binary representation, as a compact yet expressive alternative to the full-precision counterpart, has benefited tasks such as image retrieval, indexing significantly. Instead of the traditional paradigm of relying on the similarity information, e.g, pairwise similarity or triplet loss, we directly use the label information and perform discriminative learning to learn the binary representation for images. Inspired by recent success of deep learning, our proposed methods are data-driven and focus on the semantic level.

Starting from the discussion of using cross-entropy as the criterion of learning, we proposed CE-Bits, a shallow method (comparing to later end-to-end deeper approaches). By directly using deep feature as the input, it provides the best performance on retrieval and classification tasks with short training time. In order to better utilize the deep neural networks to provide better performance, we proposed the Direct Binary Embedding (DBE) layer, which can be conveniently integrated in the existing deep neural networks such as convolutional neural networks (CNNs). Meanwhile, the same learning strategy can be used (softmax cross-entropy). This simplifies the learning process and provides better performance on tasks such as retrieval, classification, and annotation.

In order to cope with multimedia data, we proposed Discriminative Cross-View Hashing (DCVH), a framework on aligning image view and textual view. As a result, the cross-retrieval between two different types of media (visual and textual content) is possible. Furthermore, it is observed that by considering the semantics of the textual information, which is usually directly used as independent labels for images, the performance of supervised image retrieval (single-view hashing) is enhanced. We also consider the scenario of multiple sources of visual data and lacking label information. To mitigate the inefficiency of direct transfer learning, we consider adopting domain adaptation into the learning of binary representation, leading to better understanding of the unlabeled domain in various visual tasks. Finally, we studied the learning of binary representation for instances in images instead of a global one. We show that such binary representation provides competitive performance on object detection. It also provides state-of-the-art performance on multi-instance retrieval task.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Man, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Vigas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 50

[2] andAndrew Zisserman, K. S. (2016). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 3, 7, 16, 17, 18, 23, 60, 70

[3] Arjovsky, M. and Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. In *ICLR*. 70

[4] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *ArXiv e-prints*. 70

[5] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3). 16, 18

[6] Bronstein, M. M., Bronstein, A. M., Michel, F., and Paragios, N. (2010). Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *2010 IEEE CVPR*. 40, 41, 50, 51

[7] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *Proc. ECCV*, pages 778–792, Berlin, Heidelberg. Springer-Verlag. 7

[8] Cao, Y., Long, M., Wang, J., Yang, Q., and Yu, P. S. (2016). Deep visual-semantic hashing for cross-modal retrieval. In *Proc. KDD*. 40, 43

[9] Cao, Z., Long, M., Wang, J., and Qiang, Y. (2017). Transitive hashing network for heterogeneous multimedia retrieval. In *Proc. AAAI*. 41, 43, 44, 50, 59, 60, 61

[10] Cao, Z., Long, M., Wang, J., and Yu, P. S. (2017). HashNet: Deep Learning to Hash by Continuation. *ArXiv e-prints*. 52, 54, 72, 79

[11] Chum, O., Philbin, J., and Zisserman, A. (2008). Near duplicate image detection: min-hash and tf-idf weighting. In *Proc. BMVC*, pages 50.1–50.10. doi:10.5244/C.22.50. 3, 7

[12] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*. 16

[13] Ding, G., Guo, Y., and Zhou, J. (2014). Collective matrix factorization hashing for multimodal data. In *2014 IEEE CVPR*, pages 2083–2090. 35, 40, 43, 50, 51, 54

[Everingham et al.] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 83

[15] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874. 17, 31

[16] Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proc. 32nd ICML*. 60, 61

[17] Ghifary, M., Kleijn, W. B., Zhang, M., Balduzzi, D., and Li, W. (2016). Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV 2016*, pages 597–613. 71

[18] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G. J. and Dunson, D. B., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings. 28

[19] Gong, Y., Jia, Y., Leung, T. K., Toshev, A., and Ioffe, S. (2014). Deep Convolutional Ranking for Multilabel Image Annotation. In *ICLR*. 36, 43, 54, 55

[20] Gong, Y. and Lazebnik, S. (2011a). Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. 3, 7, 14, 17, 18, 25, 33, 34, 35

[21] Gong, Y. and Lazebnik, S. (2011b). Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 817–824, Washington, DC, USA. 59, 72, 73

[22] Gong, Y., Pawlowski, M., Yang, F., Brandy, L., Bourdev, L., and Fergus, R. (2015). Web scale photo hash clustering on a single machine. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2, 6

[23] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 61, 67

[24] Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2006). A kernel method for the two-sample-problem. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 513–520. 61, 67

[25] Guo, R. and Qi, H. (2013). Partially-sparse restricted boltzmann machine for background modeling and subtraction. In *ICMLA*. IEEE. 3, 7, 9

[26] Hardoon, D. R., Szedmak, S. R., and Shawe-taylor, J. R. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*, 16(12). 42

[27] He, J., Liu, W., and Chang, S.-F. (2010). Scalable similarity search with optimized kernel hashing. In *Proc. KDD*, KDD '10, pages 1129–1138, New York, NY, USA. ACM. 2, 7

[28] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 80

[29] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. xiii, 3, 7, 17, 32, 33, 41, 42, 45, 60, 68

[30] Huiskes, M. J. and Lew, M. S. (2008). The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA. ACM. 49, 68

[31] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456. 28

[32] Irie, G., Arai, H., and Taniguchi, Y. (2015). Alternating co-quantization for cross-modal hashing. In *The IEEE International Conference on Computer Vision (ICCV)*. 35, 40, 41, 43, 44, 45, 50, 51, 53, 54

[33] Jaitly, N. and Hinton, G. E. (2013). Using an autoencoder with deformable templates to discover features for automated speech recognition. In *Proc. Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1737–1740. 3, 7

[34] Jiang, Q. and Li, W. (2016). Deep cross-modal hashing. *CoRR*, abs/1602.02255. 40, 41, 43, 44, 45, 50, 51, 59, 60

[35] Jiang, Q.-Y. and Li, W.-J. (2017). Deep cross-modal hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 72

[36] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*. 44

[37] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report. 32

[38] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *NIPS*. 2, 3, 6, 7

[39] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing*

Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States., pages 1106–1114. 35, 36, 50

[40] Kulis, B. and Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1042–1050. Curran Associates, Inc. 3, 7

[41] Kumar, S. and Udupa, R. (2011). Learning hash functions for cross-view similarity search. In *Proc. IJCAI*, pages 1360–1365. AAAI Press. 40, 41, 43, 44, 50, 51

[42] Lai, H., Pan, Y., Liu, Y., and Yan, S. (2015). Simultaneous feature learning and hash coding with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3, 7

[43] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 31, 32

[44] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *Proc. ICCV*. 7

[45] Lin, G., Shen, C., Shi, Q., van den Hengel, A., and Suter, D. (2014a). Fast supervised hashing with decision trees for high-dimensional data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1971–1978. 14, 17, 18, 25, 32, 33, 34

[46] Lin, K., Lu, J., Chen, C.-S., and Zhou, J. (2016). Learning compact binary descriptors with unsupervised deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7, 72, 73

[47] Lin, K., Yang, H.-F., Hsiao, J.-H., and Chen, C.-S. (2015a). Deep learning of binary hash codes for fast image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 33, 34, 36

[48] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014b). *Microsoft COCO: Common Objects in Context*, pages 740–755. Springer International Publishing, Cham. 33, 49, 68

[49] Lin, Z., Ding, G., Hu, M., and Wang, J. (2015b). Semantics-preserving hashing for cross-view retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 40, 41, 43, 44, 50, 51

[50] Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. In *ICLR 2017 (Conference Track)*. 44

[51] Liu, H., Wang, R., Shan, S., and Chen, X. (2016a). Deep supervised hashing for fast image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3, 25, 26, 34, 40

[52] Liu, L. and Qi, H. (2017). Learning effective binary descriptors via cross entropy. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE. 3, 25, 26, 40, 61

[53] Liu, L., Rahimpour, A., Taalimi, A., and Qi, H. (2017). End-to-end Binary Representation Learning via Direct Binary Embedding. *IEEE International Conference on Image Processing*. 40, 41, 43, 45, 52, 54, 55, 61, 65, 79

[54] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016b). Ssd: Single shot multibox detector. In *ECCV*. 78, 79

[55] Liu, W., Wang, J., and fu Chang, S. (2011). Hashing with graphs. In *In ICML*. 2, 7

[56] Liu, W., Wang, J., Ji, R., Jiang, Y.-G., and Chang, S.-F. (2012). Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE Computer Society. 3, 10, 14, 17, 18, 22, 25, 26, 52, 54

[57] Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. In *Proc. of the 32Nd ICML - Volume 37*, pages 97–105. 68, 71

[58] Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems 29*, pages 136–144. 60, 61, 68, 69, 71

[Luo and Qi] Luo, J. and Qi, H. Distributed object recognition via feature unmixing. In *Proc. ICDSC 2010*. ACM. 2, 7

[60] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc. 50

[61] Naikal, N., Yang, A., and Sastry, S. (2010). Towards an efficient distributed object recognition system in wireless smart camera networks. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8. 2, 6

[62] Nilsback, M.-E. and Zisserman, A. (2006). A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454. ix, 14, 16, 17, 18, 22

[63] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175. 33

[64] Ortiz, R. (2012). Freak: Fast retina keypoint. In *Proc. CVPR*, pages 510–517. 7

[65] Panareda Busto, P. and Gall, J. (2017). Open set domain adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*. 62, 72

[66] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. xiii, 41, 42, 50

[67] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151. 11

[68] Rahimpour, A., Taalimi, A., and Qi, H. (2017). Feature encoding in band-limited distributed surveillance systems. In *ICASSP 2017-IEEE International Conference on Acoustics, Speech, and Signal Proceeding*. IEEE. 25

[69] Rastegari, M., Choi, J., Fakhraei, S., Daume III, H., and Davis, L. (2013). Predictable dual-view hashing. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, page 13281336. JMLR. 35

[70] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

[71] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*. 80

[72] Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., and Zisserman, A. (2006). Using multiple segmentations to discover objects and their extent in image collections. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06. 79

[73] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proc. of the 11th ECCV: Part IV*, pages 213–226. 68, 69

[74] Salakhutdinov, R. and Hinton, G. E. (2009). Deep boltzmann machines. In *Proc. AISTATS*. 3, 7

[75] Shen, F., Shen, C., Liu, W., and Tao Shen, H. (2015). Supervised discrete hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3, 8, 9, 10, 14, 17, 18, 22, 25, 26, 31, 32, 33, 34, 40, 43, 45, 59, 61

[76] Song, J., He, T., Gao, L., Xu, X., and Shen, H. T. (2017). Deep region hashing for efficient large-scale instance search from images. *CoRR*. 79

[77] Song, J., He, T., Gao, L., Xu, X., and Shen, H. T. (2018). Deep region hashing for efficient large-scale instance search from images. 79

[78] Song, Y., Zhang, Z., Liu, L., Rahimpour, A., and Qi, H. (2016). Dictionary reduction: Automatic compact dictionary learning for classification. In *Asian Conference on Computer Vision (ACCV)*, pages 1–16. Springer. 9

[79] Sun, B. and Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In *ECCV 2016 Workshops*. 60, 61, 63, 68, 71

[80] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial Discriminative Domain Adaptation. *ArXiv e-prints*. 60, 61, 62, 71

[81] Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 62, 63, 68, 69, 71, 72

[82] Wan, L., Zeiler, M. D., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *ICML*. 3, 7

[83] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., and Xu, W. (2016). Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*. 43

[84] Weiss, Y., Torralba, A., and Fergus, R. (2009). Spectral hashing. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1753–1760. Curran Associates, Inc. 2, 3, 7, 25, 43

[85] Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., and Hao, H. (2015). Convolutional neural networks for text hashing. In *Proc. IJCAI*, pages 1369–1375. AAAI Press. xiii, 41, 42, 44, 45

[86] Yang, J. and Jiasen Lu, Dhruv Batra, D. P. (2017). A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*. 80

[87] Yao, T., Long, F., Mei, T., and Rui, Y. (2016). Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*. 3, 25, 26, 34, 59

[88] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328. 2, 60

[89] Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., and Saminger-Platz, S. (2017). Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. In *ICLR 2017 (Conference Track)*. 60, 61, 67, 68, 69, 70, 71

[90] Zhao, F., Huang, Y., Wang, L., and Tan, T. (2015). Deep semantic ranking based hashing for multi-label image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3, 7, 59, 60, 83, 85

[91] Zhao, W., Guan, Z., Luo, H., Peng, J., and Fan, J. (2017). Deep multiple instance hashing for object-based image retrieval. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3504–3510. 79, 83, 85

[92] Zhu, H., Long, M., Wang, J., and Cao, Y. (2016). Deep hashing network for efficient similarity retrieval. In *Proc. AAAI*, pages 2415–2421. 3, 25, 26, 27, 35, 40, 45, 52, 54

[93] Zhu, X., Huang, Z., Cheng, H., Cui, J., and Shen, H. T. (2013). Sparse hashing for fast multimedia search. *ACM Trans. Inf. Syst.*, 31(2):9:1–9:24. 2, 7

[94] Zhuang, B., Lin, G., Shen, C., and Reid, I. (2016). Fast training of triplet-based deep binary embedding networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 40

Appendices

A Unifying labels of MS COCO for MIRFLICKR

Table A1: "super-category" is the super category for each category provided by MS COCO; "id (COCO)" is the original category ID provided by MS COCO; "category" is the specific category name; "ordered id" is the consecutive category ID for MS COCO; "aug. MIRFLICKR id" is the corresponding augmented MIRFLICKR label ID for MS COCO images. "aug. MIRFLICKR id" = 25 accounts for the semantic that MS COCO has but MIRFLICKR does not, and vice versa. Eventually both MIRFLICKR and MS COCO are mapped into the same label semantic space, i.e., the augmented MIRFLICKR label space, which is used for evaluation of retrieving from MS COCO w.r.t. MIRFLICKR.

super-category	id (COCO)	category	ordered id	aug. MIRFLICKR id
person	1	person	1	2,7,12,14,16
vehicle	2	bicycle	2	22
vehicle	3	car	3	4,22
vehicle	4	motorcycle	4	22
vehicle	5	airplane	5	22
vehicle	6	bus	6	22
vehicle	7	train	7	22
vehicle	8	truck	8	22
vehicle	9	boat	9	22
outdoor	10	traffic light	10	25
outdoor	11	fire hydrant	11	25
outdoor	13	stop sign	12	25
outdoor	14	parking meter	13	25
outdoor	15	bench	14	25
animal	16	bird	15	3,1
animal	17	cat	16	1
animal	18	dog	17	1,6
animal	19	horse	18	1
animal	20	sheep	19	1
animal	21	cow	20	1
animal	22	elephant	21	1

Table A1 Continued.

super-category	id (COCO)	category	ordered id	aug. MIRFLICKR id
animal	23	bear	22	1
animal	24	zebra	23	1
animal	25	giraffe	24	1
accessory	27	backpack	25	25
accessory	28	umbrella	26	25
accessory	31	handbag	27	25
accessory	32	tie	28	25
accessory	33	suitcase	29	25
sports	34	frisbee	30	25
sports	35	skis	31	25
sports	36	snowboard	32	25
sports	37	sports ball	33	25
sports	38	kite	34	25
sports	39	baseball bat	35	25
sports	40	baseball glove	36	25
sports	41	skateboard	37	25
sports	42	surfboard	38	25
sports	43	tennis racket	39	25
kitchen	44	bottle	40	25
kitchen	46	wine glass	41	25
kitchen	47	cup	42	25
kitchen	48	fork	43	25
kitchen	49	knife	44	25
kitchen	50	spoon	45	25
kitchen	51	bowl	46	25
food	52	banana	47	9
food	53	apple	48	9
food	54	sandwich	49	9

Table A1 Continued.

super-category	id (COCO)	category	ordered id	aug. MIRFLICKR id
food	55	orange	50	9
food	56	broccoli	51	9
food	57	carrot	52	9
food	58	hot dog	53	9
food	59	pizza	54	9
food	60	donut	55	9
food	61	cake	56	9
furniture	62	chair	57	25
furniture	63	couch	58	25
furniture	64	potted plant	59	15
furniture	65	bed	60	25
furniture	67	dining table	61	25
furniture	70	toilet	62	25
electronic	72	tv	63	25
electronic	73	laptop	64	25
electronic	74	mouse	65	25
electronic	75	remote	66	25
electronic	76	keyboard	67	25
electronic	77	cell phone	68	25
appliance	78	microwave	69	25
appliance	79	oven	70	25
appliance	80	toaster	71	25
appliance	81	sink	72	25
appliance	82	refrigerator	73	25
indoor	84	book	74	10
indoor	85	clock	75	10
indoor	86	vase	76	10
indoor	87	scissors	77	10

Table A1 Continued.

super-category	id (COCO)	category	ordered id	aug. MIRFLICKR id
indoor	88	teddy bear	78	10
indoor	89	hair drier	79	10
indoor	90	toothbrush	80	10

Vita

Mr. Liu Liu joined Dr. Hairong Qi's lab (AICIP) in the summer of 2012. He has been working in the areas including but not limited to signal processing, computer vision, machine learning and autonomous driving. He has been publishing in many conferences including SmartGridComm, ICIP, WACV, GlobalSIP, etc. His research interests are large scale computer vision problem (indexing and retrieval), efficient neural networks, meta-learning, reinforcement learning, etc.