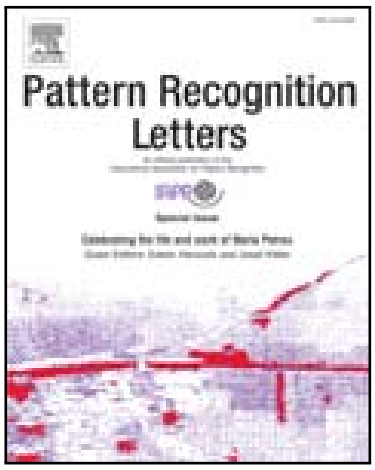


Zero-shot Hashing with Orthogonal Projection for Image Retrieval

Haofeng Zhang, Yang Long, Ling Shao

PII: S0167-8655(18)30131-4
DOI: 10.1016/j.patrec.2018.04.011
Reference: PATREC 7139

To appear in: *Pattern Recognition Letters*

Received date: 24 January 2018
Revised date: 15 March 2018
Accepted date: 7 April 2018

Please cite this article as: Haofeng Zhang, Yang Long, Ling Shao, Zero-shot Hashing with Orthogonal Projection for Image Retrieval, *Pattern Recognition Letters* (2018), doi: 10.1016/j.patrec.2018.04.011

Highlights

- Both features and attributes are projected into orthogonal space
- Binary codes of different categories are orthogonal to each other
- All the Hamming distance of different classes are equal to half code length
- Both linear and deep models are implemented to show the performance

Pattern Recognition Letters
journal homepage: www.elsevier.com

Zero-shot Hashing with Orthogonal Projection for Image Retrieval

Haofeng Zhang[a], Yang Long[b], Ling Shao[c,d,**]

[a]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[b]Open Lab, School of Computing, Newcastle University, Newcastle upon Tyne, UK
[c]Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE
[d]School of Computing Sciences, University of East Anglia, Norwich, UK

ABSTRACT

Hashing has been widely used in large-scale image retrieval. Supervised information such as semantic similarity and class label, and Convolutional Neural Network (CNN) has greatly improved the quality of hash codes and hash functions. However, due to the explosive growth of web data, existing hashing methods can not well perform on emerging images of new classes. In this paper, we propose a novel hashing method based on orthogonal projection of both image and semantic attribute, which constrains the generated binary codes in orthogonal space should be orthogonal with each other when they belong to different classes, otherwise be same. This constraint guarantees that the generated hash codes from different categories have equal Hamming distance, which also makes the space more discriminative within limited code length. To improve the performance, we also extend our method with a deep model. Experiments of both our linear and deep model on three popular datasets show that our method can achieve competitive results, specially, the deep model can outperform all the listed state-of-the-art approaches.

Keywords: Zero-shot Hashing; Orthogonal Projection; Image Retrieval

1. Introduction

There is an increasing number of researchers contribute their great endeavours on large-scale image or video retrieval due to its wide practical utilities(Bakhtiary et al., 2017; Chen et al., 2016; Lv et al., 2017; Zhang et al., 2018). Hashing is a powerful and well-known large-scale image retrieval technique, which encode the real-valued image into binary codes, such as *'0/1'* or *'-1/+1'*. Processed by hashing methods, large-scale image dataset can be compressed with small storage cost and retrieved efficiently using Hamming distance or bit-wise XOR operation.

The earlier hashing methods are data-independent, such as Locality-sensitive hashing (LSH) (Gionis et al., 1999) which hashes input items so that similar items map to the same bucket with high probability, but it is verified that LSH cannot get satisfactory results in many scenarios. There are two categories of data dependent hashing methods, include unsupervised hashing such as Anchor Graph Hashing (AGH) (Liu et al., 2011) and supervised hashing like Canonical Correlation Analysis based Iterative Quantization (CCA-ITQ) (Gong et al., 2013). With the auxiliary information like category label or semantic similarity, supervised hashing often can achieve excellent performance because supervising information can assist to explore the intrinsic property in the training data.

Earlier Hashing methods often use handcraft features such as GIST (Liu et al., 2017b) or SIFT (Carreira-Perpinán and Raziperchikolaei, 2015), which can achieve great improvement, but still cannot meet our demands on many applications. Fortunately, at near recent, due to the great development of deep learning, hashing methods begin to extract deep features via Convolutional Neural Network (CNN), which inherit high discriminative property and often get state-of-the-art performance, e.g. Deep Hahing (DH) (Erin Liong et al., 2015) and Deep Binary Descriptor (Deepbit) (Lin et al., 2016) directly apply CNN layers in their models to improve the performance, and BDNN (Do et al., 2016) exploits pre-extracted deep features as its train-

**Corresponding author: Tel.: +44-1603-59-2603 fax: +44-1603-59-2603;
e-mail: zhanghf@njust.edu.cn (Haofeng Zhang),
yang.long@ieee.org (Yang Long), ling.shaog@ieee.org (Ling Shao)

ing input.

We should have noticed that the conventional supervised hashing methods usually require the query data and the training data must have the same distribution, means that the classes of query data should appear in the training classes. However, along with the explosive growth of web data, the requirement is becoming increasingly difficult to be satisfied because the images of new semantic concepts are emerging rapidly. It is costly to label sufficient training data for the new emerged semantic categories, and unrealistic to retrain the hash function when new concept appears. Therefore, it is necessary to find new hashing methods to solve the retrieval problem of the new classes.

Zero-shot Recognition (ZSR) (Long et al., 2017b, 2016, 2017a) aims to learning a classification model which is trained on the samples belong to seen classes, but can be transferred to be applied on test data belongs to unseen classes. In ZSR, seen and unseen classes are usually related in a high dimensional vector space, which is called semantic embedding space. Such a space is often an attribute space or a word vector space. Therefore, to solve the hashing problem of new classes, the best way is to adopt the technique of semantic embedding. As illustrated in figure (1), if we do not have the class *'wildebeest'* in training set, but we have the description in *'Wikipedia'*: "*'wildebeest'* is a large antelope living in the African steppe, the head of the wildebeest is thick and shoulder-wide, like a buffalo; the rear is slender, more horse-like".[1] Hence, the properties of *'antelope'*, *'buffalo'* and *'horse'* can be transferred to *'wildebeest'* to build a hash function.

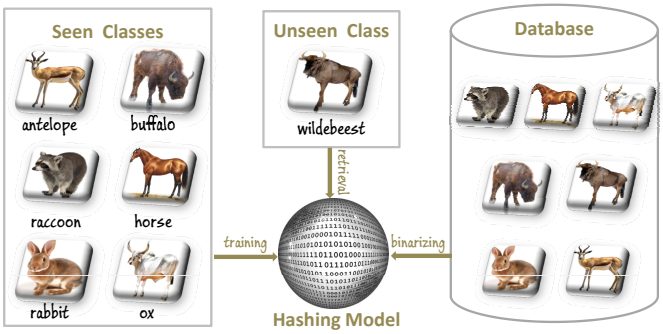


Fig. 1. Framework of Zero-shot Hashing, which is trained only on seen classes, but should be well generalised on unseen classes.

Y. Yang *et al.* (Yang et al., 2016) utilised the advantages of ZSR to solve the problem of zero-shot image retrieval, and proposed a method called Zero-shot Hashing (ZSH) via Transferring Supervised Knowledge (TSK), which exploits Discrete Supervised Hashing (DSH) to generate binary codes, and then projects the generated binary codes into semantic embedding space, and force these codes to have their original semantic concepts. SitNet (Guo et al., 2017) is another ZSH method, which takes advantages of powerful feature extraction ability of CNN, and construct a loss function including center loss and max-margin loss to optimise the network model. However, the center loss and the max-margin loss in SitNet play similar role in enlarging the distance between classes. Moreover, because of the limitation of binary code length, the max-margin loss also have the problem that it might make some classes have large distance while some have very small distance, which often leads to discriminative confusion.

In this paper, we propose an orthogonal projection based Zero-shot Hashing model, which can avoid the disadvantage of max-margin loss. The propose method projects both image and semantic attribute into binary orthogonal space, and obliges them to be orthogonal with each other if they belong to different categories, otherwise to be same. This constraint can guarantee that the generated binary vectors of different semantic concepts have equal Hamming distance, which is half the code length. The orthogonal constraint is better than the max-margin loss, because it can create more discriminative semantic binary space to alleviate the confusion. Moreover, to exploit the powerful tool of deep learning, we extend our method with a deep model, which might greatly improve the performance. In summary, we make the following contributions in this paper,

1) We propose a novel method base on orthogonal projection to solve the Zero-shot Hashing problem, which can generate effective binary codes for the unseen classes by transferring the semantic concepts of the seen classes.
2) We build a Zero-shot Hashing framework, which takes both images and semantic attribute as input and generate their corresponding binary codes, which is obliged to be orthogonal with each other when they belong to same category, otherwise to be same. This constraint force the generated binary code of different classes to have equal Hamming distance, which can guarantee more discriminative space within limited code length.
3) We extend our method with a deep model, which utilises the powerful non-linear fitting ability of deep learning. With the deep model, our method can achieve great improvement.

The rest of this paper is organized as follows. In Section 2, we give a brief review of recent methods on Hashing, Zero-shot Learning and Zero-shot Hashing. The details of our method for linear projection and deep projection are both described in Section 3. Section 4 reports the experimental results of our method on three popular datasets. Finally, we conclude this paper in Section 5.

2. Related Works

In this section, we briefly review some related works on hashing, zeros-shot learning and zero-shot hashing.

2.1. Hashing

Hashing is one of similarity searching techniques, it encodes images (Liu et al., 2017c,a), videos (Wu et al., 2017) *etc.* into binary codes for low-cost storage and effective searching. One of the earliest hashing method is LSH (Gionis et al., 1999), which is data-independent and differs from conventional and cryptographic hash functions because it aims to maximize the

[1] https://en.wikipedia.org/wiki/Wildebeest

probability of a collision for similar items. But this type of methods can not obtain satisfactory performance on retrieval tasks, thence data-dependent approaches are proposed to solve this problem.

Data-dependent methods can be classified into two categories: unsupervised and supervised hashing. Unsupervised hashing methods exploit statistical or graph based approaches to mine the latent properties in the data, which is used to optimise the hash functions, such as K-Means Hashing (KMH) (He et al., 2013), which performs k-means clustering and learns the binary indices of the quantized cells. Graph based unsupervised hashing methods like AGH (Liu et al., 2011) and Inductive Manifold Hashing (IMH) (Shen et al., 2013) usually construct graphs from training images and optimise hash functions by preserving the graph structures. Supervised hashing use the labels of training data as guiding information to optimise hash function, and greatly improve the performance comparing to unsupervised hashing. Supervised hashing techniques has been emerging continuously in recent years, representative methods include Supervised Discrete Hashing (SDH) (Shen et al., 2015; Gui et al., 2017), Kernelised Supervised Hashing (KSH) (Liu et al., 2012), *etc.* Recently, with the success of deep learning, many researchers turn to adopt CNN into hashing. DH (Erin Liong et al., 2015) and Deepbit (Lin et al., 2016) are two representative unsupervised deep hashing methods. Supervised deep hashing algorithm like Deep Supervised Hashing (DSH) (Liu et al., 2016), Supervised Semantic-preserving Deep Hashing (SSDH) (Yang et al., 2017) can achieve competitive results.

2.2. Zero-shot Learning

Since visual attributes (Ferrari and Zisserman, 2008) has been proposed, extensive researches (Kankuekul et al., 2012; Lampert et al., 2009) have been conducted on how to learn intermediate attribute classifiers for zero-shot learning tasks. According to the ways of using the features and attributes, we simply divide the methods into four categories, including Compatibility Learning, Transductive Learning and Synthetic Learning.

For the first category, Compatibility Learning framework learns linear or non-linear projection from feature space to attribute space or latent space by using only seen features and attributes, and then is applied on unseen features. These methods include linear models like Direct Attribute Prediction (DAP) (Lampert et al., 2014), Deep Visual Semantic Embedding (DEVISE) (Frome et al., 2013), Attribute Label Embedding (ALE) (Akata et al., 2016), Structured Joint Embedding (SJE) (Akata et al., 2015), Semantic Auto-Encoder (SAE) (Kodirov et al., 2017), and non-linear models, such as Latent Embedding (LATEM) (Xian et al., 2016), and Semantically Consistent Regularization (SCoRe) (Morgado and Vasconcelos, 2017). For the second category, Transductive Learning (Fu et al., 2014; Guo et al., 2016) is a new proposed research direction of ZSL, which postulates that in zero-shot learning problem seen class source including features and their corresponding attributes are provided, also the unlabelled target domain data is collected for learning a mapping function. Although transductive learning can greatly reduce the domain shift problem, the setting of it differs from the original purpose of zero-shot learning because the target domain data should be strictly inaccessible. The last category, Synthetic learning (Zhang et al., 2017; Lu et al., 2017; Long et al., 2017b) is a new type of method for zero-shot learning, which synthesise new features or new models from original semantic embeddings, and then use conventional classifiers such as SVM, LDA to train a model.

ZSL related methods often rely on the intermediate attributes, which represent the semantic embeddings of both seen and unseen classes. Conventional attributes (Huang et al., 2015) are high dimensional, and usually annotated by experts with real values, this type of annotation need expert knowledge, and cost a lot of manpower. To solve this problem, some methods (Al-Halah and Stiefelhagen, 2017) turn to use Word2Vec (Mikolov et al., 2013) to generate attributes based on the dataset *'Wikipedia'*. However, the textual description of the *'Wikipedia'* might be very noisy and not directly related to the visual appearance, which often lead to great degradation of performance. Another semantic attribute representation is based on similarity, which can be annotated by humans (Yu et al., 2013) or textual vectors (Demirel et al., 2017).

2.3. Zero-shot Hashing

ZSH utilised the advantages of semantic embeddings to solve the problem of zero-shot image retrieval. As we have known, there are only two methods designed directly for ZSH, which are TSK (Yang et al., 2016) and SitNet (Guo et al., 2017). TSK exploits SDH (Shen et al., 2015) to generate binary codes, and then projects the generated binary codes into semantic embedding space, and force these codes to have their original semantic concepts. SitNet adopts CNN into its framework, and construct a loss function including center loss and max-margin loss to optimise the network model. However, the center loss and the max-margin loss in SitNet play similar role in enlarging the distance between classes. Moreover, because of the limitation of binary code length, the max-margin loss also have the problem that it might make some classes have large distance while some have very small distance, which often leads to discriminative confusion.

3. Methodology

3.1. Problem Definition

Let $\boldsymbol{Y} = \{y_1, \cdots, y_s\}$ and $\boldsymbol{Z} = \{z_1, \cdots, z_u\}$ denote a set of s seen and u unseen class labels, and they are disjoint $\boldsymbol{Y} \cap \boldsymbol{Z} = \emptyset$. Similarly, let $\boldsymbol{A_Y} = \{\boldsymbol{a_{y1}}, \boldsymbol{\cdots}, \boldsymbol{a_{ys}}\} \in \mathbb{R}^{d_A \times s}$ and $\boldsymbol{A_Z} = \{\boldsymbol{a_{z1}}, \boldsymbol{\cdots}, \boldsymbol{a_{zu}}\} \in \mathbb{R}^{d_A \times u}$ denote the corresponding s seen and u unseen attributes respectively. Given the training data in 3-tuple of N seen samples: $(\boldsymbol{x}_1, \boldsymbol{a}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{a}_N, \boldsymbol{y}_N) \subseteq \boldsymbol{X}_s \times \boldsymbol{A_Y} \times \boldsymbol{Y}$, where $\boldsymbol{X}_s$ is N seen images and the preliminary knowledge for testing is u pairs of attributes and their corresponding labels:$(\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{z}}_1), \cdots, (\hat{\boldsymbol{a}}_u, \hat{\boldsymbol{z}}_u) \subseteq \boldsymbol{A_Z} \times \boldsymbol{Z}$. Zero-shot Hashing aims to learn a hash function $\mathcal{H}(\boldsymbol{x})$: $\boldsymbol{x}_i \rightarrow \{-1, 1\}^\ell$ to compute the binary code of the input image from both seen and unseen classes, where ℓ is the selected binary code length, $\boldsymbol{x}_i \in \boldsymbol{X}_s \cup \boldsymbol{X}_u$ and $\boldsymbol{X}_u$ is unseen samples, which is totally unavailable during training. The learned hash function $\mathcal{H}(\boldsymbol{x})$ should

not only guarantee that the generated binary codes of same category have short Hamming distance, but also generalise well on unseen classes.

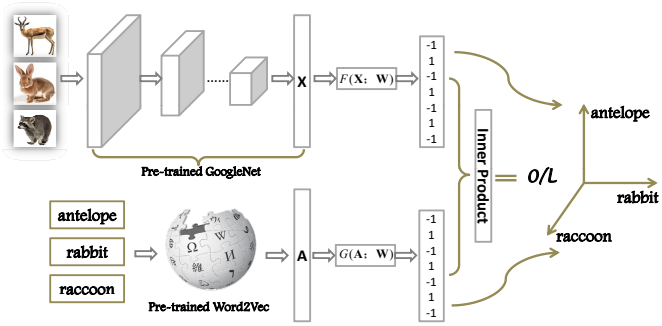


Fig. 2. Framework of our method.

3.1.1. Framework of Our Method

The training framework of our method is illustrated in figure (2). There are two branches in our architecture: the upper part is the hash function $\mathcal{H}$, which generates binary codes from images, and the bottom part is the mapping function $\mathcal{K}$ which computes binary codes from class names. When the input image and the input class name belong to same class, the generated binary code should be same with each other, otherwise, they should be orthogonal to each other. Hence, the inner product of the generated binary codes should be ℓ for same category and 0 for different categories. In the upper part hash function $\mathcal{H}$, we adopt a pre-trained GoogleNet (Szegedy et al., 2015) to extract features from input images, and in the bottom part mapping function $\mathcal{K}$, we utilise the Word2Vec (Mikolov et al., 2013) to generate class attributes from category names. For the sake of simplicity, we replace the hash function $\mathcal{H}$ with a non-linear function $F(\boldsymbol{X};\boldsymbol{W})$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \in \mathbb{R}^{N\times d_X}$ is the extracted features from images, and replace the mapping function $\mathcal{K}$ with another non-linear function $G(\boldsymbol{A};\boldsymbol{W})$, where $\boldsymbol{A} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_C\} \in \mathbb{R}^{C\times d_A}$.

3.2. Linear Model

3.2.1. Objective

Given an input feature matrix $\boldsymbol{X}$ and an input attribute matrix $\boldsymbol{A}$, according to the framework in figure (2), suppose the output of the hash function $F(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}_1$ is $\boldsymbol{B}_1$, and the output of the mapping function $G(\boldsymbol{A}) = \boldsymbol{A}\boldsymbol{W}_2$ is $\boldsymbol{B}_2$. Then the linear loss function can be designed as,

$$\begin{aligned}\min_{\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{B}_1,\boldsymbol{B}_2} &\|\boldsymbol{X}\boldsymbol{W}_1 - \boldsymbol{B}_1\|_F^2 + \|\boldsymbol{A}\boldsymbol{W}_2 - \boldsymbol{B}_2\|_F^2 + \alpha\|\boldsymbol{X}\boldsymbol{W}_1(\boldsymbol{A}\boldsymbol{W}_2)^T - \ell\boldsymbol{S}\|_F^2 \\ &+ \gamma\|\boldsymbol{W}_1\|_F^2 + \kappa\|\boldsymbol{W}_2\|_F^2 \\ s.t.\quad &\boldsymbol{B}_1 = \boldsymbol{B}_2, \boldsymbol{B}_1 \in \{-1,1\}^{n\times\ell}, \boldsymbol{B}_2 \in \{-1,1\}^{n\times\ell},\end{aligned} \tag{1}$$

where, $\|\cdot\|_F^2$ is the Frobenius norm, and $\boldsymbol{B}_1 = \boldsymbol{B}_2$ is the constraint that guarantee the generated binary codes should be same from both features and attributes. The third term constrains that the binary codes generated from the feature and the attribute which belong to same category should be same, otherwise should be orthogonal, and $\boldsymbol{S}$ is a similarity matrix, where $s_{ij} = 1$ if the feature $\boldsymbol{x}_i$ and $\boldsymbol{a}_j$ came from same category, otherwise $s_{ij} = 0$. α, γ, κ are three balancing coefficients.

To solve the dependency $\boldsymbol{B}_1 = \boldsymbol{B}_2$, we exploit a Lagrange multiplier and rewrite the equation (1) as,

$$\begin{aligned}\min_{\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{B}_1,\boldsymbol{B}_2} &\|\boldsymbol{X}\boldsymbol{W}_1 - \boldsymbol{B}_1\|_F^2 + \|\boldsymbol{A}\boldsymbol{W}_2 - \boldsymbol{B}_2\|_F^2 + \alpha\|\boldsymbol{X}\boldsymbol{W}_1(\boldsymbol{A}\boldsymbol{W}_2)^T - \ell\boldsymbol{S}\|_F^2 \\ &+ \beta\|\boldsymbol{B}_1 - \boldsymbol{B}_2\|_F^2 + \gamma\|\boldsymbol{W}_1\|_F^2 + \kappa\|\boldsymbol{W}_2\|_F^2 \\ s.t.\quad &\boldsymbol{B}_1 \in \{-1,1\}^{n\times\ell}, \boldsymbol{B}_2 \in \{-1,1\}^{n\times\ell},\end{aligned} \tag{2}$$

where, β is a balancing coefficient same as α, γ, κ.

3.2.2. Optimisation

The equation (2) has four variables $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{B}_1, \boldsymbol{B}_2$, and the function is a non-convex problem with the binary constraints. As we have known that there is no direct method to solve this problem. Therefore, we solve it via iterative optimising one variable with other variables fixed.

$\boldsymbol{B}_2$ step: With $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{B}_1$ fixed, we can get,

$$\begin{aligned}\min_{\boldsymbol{B}_2} &\|\boldsymbol{A}\boldsymbol{W}_2 - \boldsymbol{B}_2\|_F^2 + \beta\|\boldsymbol{B}_1 - \boldsymbol{B}_2\|_F^2 \\ s.t.\quad &\boldsymbol{B}_2 \in \{-1,1\}^{n\times\ell}.\end{aligned} \tag{3}$$

Equation (3)can be easily written to,

$$\begin{aligned}\min_{\boldsymbol{B}_2} &-tr(\boldsymbol{B}_2^T(\boldsymbol{A}\boldsymbol{W}_2 + \beta\boldsymbol{B}_1)) \\ s.t.\quad &\boldsymbol{B}_2 \in \{-1,1\}^{n\times\ell},\end{aligned} \tag{4}$$

which has the optimal solution,

$$\boldsymbol{B}_2 = sgn(\boldsymbol{A}\boldsymbol{W}_2 + \beta\boldsymbol{B}_1). \tag{5}$$

$\boldsymbol{B}_1$ step: With $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{B}_2$ fixed, we can get,

$$\begin{aligned}\min_{\boldsymbol{B}_1} &\|\boldsymbol{X}\boldsymbol{W}_1 - \boldsymbol{B}_1\|_F^2 + \beta\|\boldsymbol{B}_1 - \boldsymbol{B}_2\|_F^2 \\ s.t.\quad &\boldsymbol{B}_1 \in \{-1,1\}^{n\times\ell},\end{aligned} \tag{6}$$

which can be easily rewritten as,

$$\begin{aligned}\min_{\boldsymbol{B}_1} &-tr(\boldsymbol{B}_1^T(\boldsymbol{X}\boldsymbol{W}_1 + \beta\boldsymbol{B}_2)) \\ s.t.\quad &\boldsymbol{B}_1 \in \{-1,1\}^{n\times\ell}.\end{aligned} \tag{7}$$

it is readily solved as,

$$\boldsymbol{B}_1 = sgn(\boldsymbol{X}\boldsymbol{W}_1 + +\beta\boldsymbol{B}_2). \tag{8}$$

$\boldsymbol{W}_1$ step: With $\boldsymbol{W}_2$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ fixed, we can get,

$$\min_{\boldsymbol{W}_1} \|\boldsymbol{X}\boldsymbol{W}_1 - \boldsymbol{B}_1\|_F^2 + \alpha\|\boldsymbol{X}\boldsymbol{W}_1(\boldsymbol{A}\boldsymbol{W}_2)^T - \ell\boldsymbol{S}\|_F^2 + \gamma\|\boldsymbol{W}_1\|_F^2. \tag{9}$$

We derivative Eq. (9) with regard to $\boldsymbol{W}_1$, and then set it to zero, we can get the following formulation,

$$(\boldsymbol{X}\boldsymbol{X}^T + \gamma\boldsymbol{I})\boldsymbol{W}_1 + \alpha\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W}_1(\boldsymbol{A}\boldsymbol{W}_2)^T\boldsymbol{A}\boldsymbol{W}_2 = \boldsymbol{X}^T\boldsymbol{B}_1 + \alpha\ell\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{A}\boldsymbol{W}_2. \tag{10}$$

We define $\widehat{\boldsymbol{A}}_1 = (\alpha \boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X} + \gamma\boldsymbol{I})$, $\widehat{\boldsymbol{B}}_1 = (\boldsymbol{A}\boldsymbol{W}_2)^T\boldsymbol{A}\boldsymbol{W}_2$, $\widehat{\boldsymbol{C}}_1 = (\alpha \boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{B}_1 + \alpha\ell\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{A}\boldsymbol{W}_2)$, then the equation (10) can be rewritten as,

$$\widehat{\boldsymbol{A}}_1\boldsymbol{W}_1 + \boldsymbol{W}_1\widehat{\boldsymbol{B}}_1 = \widehat{\boldsymbol{C}}_1. \tag{11}$$

Equation (11) is a well-know Sylvester equation which can be solved efficiently by the Bartels-Stewart algorithm(Bartels and Stewart, 1972). In Matlab, it can be solved by using only one line code $W_1 = sylvester(\widehat{A}_1, \widehat{B}_1, \widehat{C}_1)$[2].

W_2 step: With $\boldsymbol{W}_1$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ fixed, we can get,

$$\min_{\boldsymbol{W}_2} \|\boldsymbol{A}\boldsymbol{W}_2 - \boldsymbol{B}_2\|_F^2 + \alpha\|\boldsymbol{X}\boldsymbol{W}_1(\boldsymbol{A}\boldsymbol{W}_2)^T - \ell\boldsymbol{S}\|_F^2 + \kappa\|\boldsymbol{W}_2\|_F^2. \tag{12}$$

We derivative Eq. (12) with regard to $\boldsymbol{W}_1$, and then set it to zero, we can get the following formulation,

$$(\boldsymbol{A}\boldsymbol{A}^T + \kappa\boldsymbol{I})\boldsymbol{W}_2 + \alpha\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{W}_2(\boldsymbol{X}\boldsymbol{W}_1)^T\boldsymbol{X}\boldsymbol{W}_1 = \boldsymbol{A}^T\boldsymbol{B}_2 + \alpha\ell\boldsymbol{A}^T\boldsymbol{S}^T\boldsymbol{X}\boldsymbol{W}_1 \tag{13}$$

Similarly, we define $\widehat{\boldsymbol{A}}_2 = (\alpha\boldsymbol{A}^T\boldsymbol{A})^{-1}(\boldsymbol{A}^T\boldsymbol{A} + \kappa\boldsymbol{I})$, $\widehat{\boldsymbol{B}}_2 = (\boldsymbol{X}\boldsymbol{W}_1)^T\boldsymbol{X}\boldsymbol{W}_1$, $\widehat{\boldsymbol{C}}_2 = (\alpha\boldsymbol{A}^T\boldsymbol{A})^{-1}(\boldsymbol{A}^T\boldsymbol{B}_2 + \alpha\ell\boldsymbol{A}^T\boldsymbol{S}^T\boldsymbol{X}\boldsymbol{W}_1)$, then the equation (13) can be rewritten as,

$$\widehat{\boldsymbol{A}}_2\boldsymbol{W}_2 + \boldsymbol{W}_2\widehat{\boldsymbol{B}}_2 = \widehat{\boldsymbol{C}}_2. \tag{14}$$

Similar as the equation (11), we can get the result using $W_2 = sylvester(\widehat{A}_2, \widehat{B}_2, \widehat{C}_2)$.

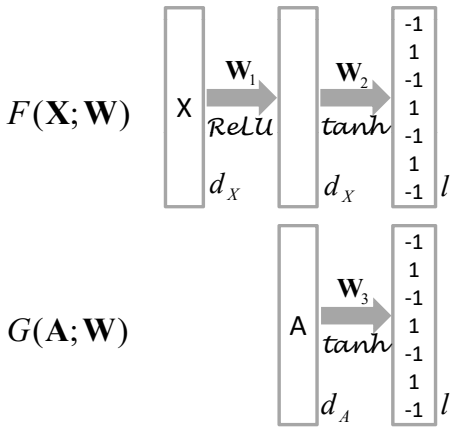


Fig. 3. Architecture of deep layers.

3.3. Deep Orthogonal Projection Network

To obtain better hash function, we adopt the deep structure to extend our linear model into deep model, which is illustrated in figure (3). In our deep model, the hash function utilise three full connection layers, which are $d_X \rightarrow d_X(ReLU) \rightarrow \ell(tanh)$, and the mapping function exploits two full connection layers, which are $d_A \rightarrow \ell(tanh)$.

Given an input feature $\boldsymbol{x}_i$ extracted from image I_i, and an input attribute $\boldsymbol{a}_j$ generated from class name t_j, suppose the output of hash function $F(\boldsymbol{x}_i)$ is $\boldsymbol{b}_i \in \{-1, 1\}^\ell$, and the output of mapping function $G(\boldsymbol{a}_j)$ is $\boldsymbol{b}_j \in \{-1, 1\}^\ell$. When the feature $\boldsymbol{x}_i$ and the attribute $\boldsymbol{a}_j$ belong to same category, we should oblige the corresponding generated binary code to be same, which also means that the inner product of them should be the code length ℓ, otherwise, if they come from different category, we force the Hamming distance of the generated binary codes should be half of code length, which also means that the inner product of them should be 0. Here we do not use the max-margin loss to force the binary codes of different classes to have largest Hamming distance, because the code length is limited, which may lead to bad localisation that parts of categories have large distance in Hamming space and some of them have similar binary codes. Therefore, by mixing the features and attributes and constructing the training pairs, the the loss function can be designed as,

$$\min_{\boldsymbol{W}} -\frac{1}{K}\sum_{k=1}^{K}(s_k \ln\frac{P_k}{\ell} + (1 - s_k)\ln(1 - \frac{P_k}{\ell})), \tag{15}$$

where, K is the number of training samples, P_k is the inner product of $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, and $s_k \in \{0, 1\}$ is similarity of $\boldsymbol{x}_i$ and $\boldsymbol{a}_j$.

3.4. Training Pairs Sampling

In this paper, we need not only the similar pairs which include features and their corresponding attributes, but also the dissimilar pairs which contain features and attributes that they belong to different classes, *e.g.* a feature belongs to the type of *Chimpanzee* and an attribute belongs to the type of *Chimpanzee* construct a similar pair, and a feature of *Chimpanzee* and an attribute of *Leopard* are made of a dissimilar pair. In our linear model, we construct the similar matrix $\boldsymbol{S} = \{0, 1\}^{N\times N}$, where if the feature and the attribute are similar, the entry of S is $s_{ij} = 1$, else $s_{ij} = 0$. In our deep orthogonal projection network, we set the input as 3-tuple vector $(\boldsymbol{x}_i, \boldsymbol{a}_i, s_i)$, where if the feature $\boldsymbol{x}_i$ and the attribute $\boldsymbol{a}_j$ belong to same class, then s_k is set to 1, otherwise set to 0. At each epoch, similar pairs are selected using all the features and their corresponding attributes, thus there are N similar pairs; dissimilar pairs are made of all the features and randomly picked dissimilar attributes, which also forms N dissimilar pairs, thus we have $K = 2N$ pairs in each epoch. Besides, at the beginning of each epoch, we regenerate the dissimilar pairs and shuffle the K input pairs.

4. Experiments

4.1. Datasets and Semantic Embeddings

In our experiments, we employ three popular datasets, including Animals with Attributes (AWA) (Lampert et al., 2014), CIFAR-10 (Krizhevsky, 2009), and ImageNet (Deng et al., 2009).

AWA Dataset AWA is a coarse-grained and medium-scale dataset, which contains 30,475 images coming from 50 animal categories. This dataset is widely used for Zero-shot Learning (ZSL).

CIFAR-10 CIFAR-10 contains 10 non-overlapping classes and each class consists of 6,000 images, leading to a total of 60,000 images, and each image has the size of 32×32. CIFAR-10 is often used for evaluating hashing models.

[2]https://uk.mathworks.com/help/matlab/ref/sylvester.html

ImageNet ImageNet is a large scale vision dataset, which is organised according to the WordNet (Miller, 1995) hierarchy. In this experiment, we utilise the subset of ImageNet for the Large Scale Visual Recognition Challenge (ILSVRC2012), which has 1K categories, and totally about 1.3M images. ImageNet is often used as large scale dataset for evaluating visual recognition models.

Semantic Embeddings Semantic embeddings play an important role in zero-shot related learning methods, it transfer knowledge from seen classes to unseen classes. In our experiments, we exploit the Word2Vec (Mikolov et al., 2013) to extract 300 dimensional semantic vectors from class names.

4.2. Settings

We adopt the same settings as that described in (Yang et al., 2016; Guo et al., 2017) for all the three datasets. For dataset AWA, we split all the 50 classes into 5 groups randomly, and each group has 10 classes. We use one group as the unseen concepts and other four groups are set as the seen concepts, thence we have 5 different splits. For dataset CIFAR-10, we randomly select one category as unseen class and the left nine classes are set as seen classes for training, thence we can obtain 10 different splits. For the largest dataset ImageNet, we randomly select 100 categories form the total 1K categories, and can get about 130,000 images. We randomly pick 10 categories as the unseen concepts and the left 90 classes are set as the seen concepts.

For all the three datasets, we randomly choose 10,000 images from the seen classes as the training set, and randomly pick 1,000 images from the unseen classes as the query set. The left unseen images and all the seen images are merged to construct the retrieval database, *e.g.* for dataset AWA, 1000 images from 10 unseen classes are set as the query set, and the left 29,475 images are set as the retrieval database.

We implement both linear and deep models of our method, and compare them with eight state-of-the-art method, including Discrete Similarity Transfer Network (SitNet) (Guo et al., 2017), Iterative Quantisation (ITQ) (Gong et al., 2013), Inductive Manifold Hashing (IMH) (Shen et al., 2013), Kernelised Supervised Hashing (KSH) (Liu et al., 2012), Supervised Discrete Hashing (SDH) (Shen et al., 2015), Deep Hashing Network (DHN) (Zhu et al., 2016), Deep Neural Network Hashing (DNNH) (Lai et al., 2015), and Transferring Supervised Knowledge (TSK) (Yang et al., 2016). In these eight methods, ITQ and IMH are unsupervised hashing approaches, SDH and KSH are two supervised methods, DHN, DNNH, and SitNet are three CNN based methods, TSK and SitNet are the only two methods completely designed for Zero-shot Hashing.

We adopt the two popular evaluation metrics, mean Average Precision (mAP) and Precision within Hamming radius 2 (Precision@r2), in our experiments. The results for these two metrics in our experiments are directly cited from (Guo et al., 2017), and the retrieved results on CIFAR-10 of the eight methods are implemented by ourselves or using the codes provided by the authors or the others.

In our experiments, for the sake of simplicity, we use the extracted features with the pre-trained GoogleNet model (Szegedy et al., 2015). In linear model, we set $\alpha = \beta = 1, \gamma = \kappa = 0.0001$, and to avoid the singularity of $\boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{A}^T\boldsymbol{A}$, we add an auxiliary matrix $\omega\boldsymbol{I}$ to them, where $\omega = 0.0001$. In our deep model, the learning rate is set as 10^{-4}, the training batch size is set as 20, and the training iteration is 2×10^5.

4.3. Comparison

4.3.1. Retrieval Precision

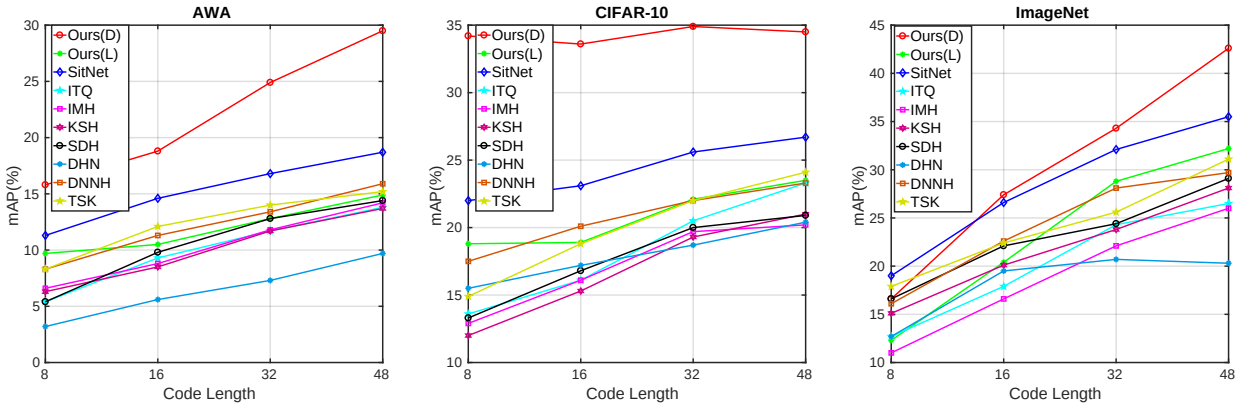
All the comparing methods are implemented on four different code lengths, 8 bits, 16 bits, 32 bits and 48 bits. Since there are 10 splits for datasets AWA and CIFAR-10, we compute the metrics of mAP and Precision@r2 using the 10 splits, and then record the average results of them. For dataset ImageNet, we firstly generate 10 different splits randomly, and then execute our method on each split and obtain 10 different performances, which are averaged and recorded as the final result at last. We plot the mAP in figure (4) and present the Precision@r2 in figure (5). It can be noticed that our method of deep model almost outperforms all the baseline methods mentioned above. Besides, we also have the following observations.

In figure (4) and figure (5), the performance of the two supervised methods, DSH and KSH, degraded greatly comparing to the performance on seen classes, which might be caused by the fact that the supervised information does not include the unseen classes, and lead to over fitting on the unseen classes. on the contrary, the unsupervised methods do not need supervised information, so the performances do not reduce too much. The deep network based methods like DNNH and SitNet performs better than most of the listed approaches, which is due to the powerful fitting ability of deep network. SitNet and TSK get better results than almost all of the remaining methods except ours, because these two methods are designed using the semantic embeddings, and their purpose is to transfer their model to fit the unseen classes.

For the metric of mAP, our linear model achieves similar performance as TSK, and a little worse than SitNet, which is deep network based method and has great fitting ability. For the metric of Precision@r2, our linear model perform bad on short code length, but can obtain significant improvement on long code length, and can outperform all the listed eight methods.

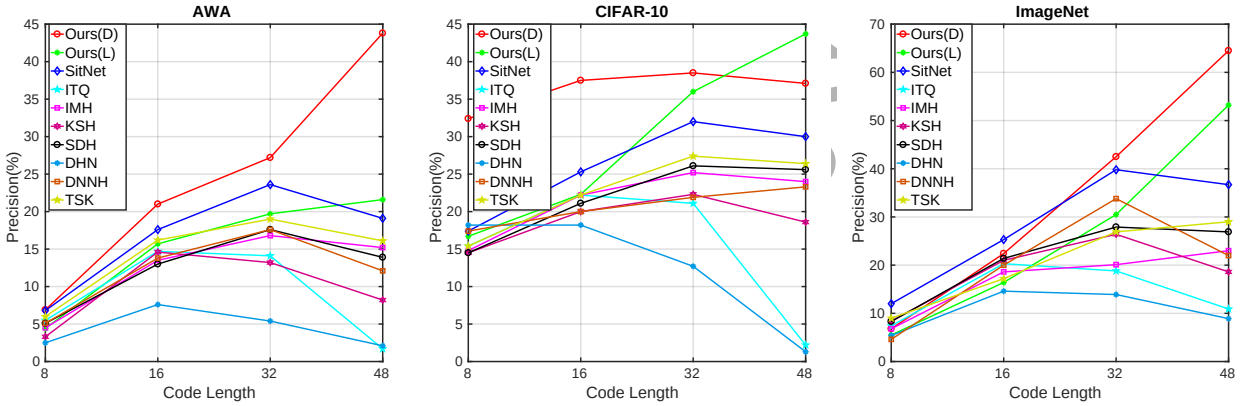
Our deep model can outperform all the eight methods on datasets AWA and CIFAR-10, and can win at 32 bits and 48 bits on dataset ImageNet for both mAP and Precision@r2. The Precision@r2 curves of all the listed eight methods declines when the code length is longer than 32 bits, while our method of both linear and deep models can still keep the upward trend on dataset AWA and ImageNet. The Precision@r2 on dataset CIFAR-10 has a little drop when the code length is 48 bits comparing to the performance on 32 bits.

The performance of our deep methods can exceed a lot on all the four types of code length on dataset CIFAR-10. On dataset AWA and ImageNet, our deep model surpass the bigger gap when the code length is longer. This phenomenon is caused by the category numbers of the datasets, *e.g.* the CIFAR-10 has 10 categories, which is less than most the code lengths, thence there is enough space to place all the semantic binary vectors of all the 10 classes. While AWA has 50 categories, which is longer than all the code lengths, thence when the code length



(a) AWA (b) CIFAR-10 (c) ImageNet

Fig. 4. Mean Average Precision (mAP) on three popular datasets. Ours(D) and Ours(L) represent the deep model and the linear model of our method respectively.



(a) AWA (b) CIFAR-10 (c) ImageNet

Fig. 5. Precision with Hamming radius 2 on three popular datasets. Ours(D) and Ours(L) represent the deep model and the linear model of our method respectively.

is increasing, the orthogonal space can be placed with more orthogonal semantic binary vectors.

4.3.2. Retrieved Images

To show the performance of our method more intuitively, we randomly select one query image as example from the class *'truck'* of the dataset CIFAR-10, and exhibit the top twenty retrieved images by all the listed methods and our deep model on code length 48 bits. Additionally, we also computed the corresponding mAPs of the returned 20 images for all the methods. It can be seen that the proposed method can achieve more satisfying results and get higher mAP comparing to the listed other methods. Among the retrieved images, our method gets eight false-positive images, which are least among all the listed approaches. Besides, our method can get six correct results of the top ten retrieved images, which are also the most correct returned images. The false-positive returned images of our method mainly lie in the class *'automobile'*, which is the most similar word as *'truck'* in all the nine class names. Such failure cases also support that our resultant binary codes are learned using the knowledge of the seen classes. In comparison, the reason behind the failure cases of other conventional methods such as KSH and ITQ are easier to be found out, *e.g. 'frog'* is a false-positive of *'truck'* because they have the similar color feature. Our method utilises the semantic embeddings and constructs enough orthogonal vector space, which must be the reason why it can outperform other methods.

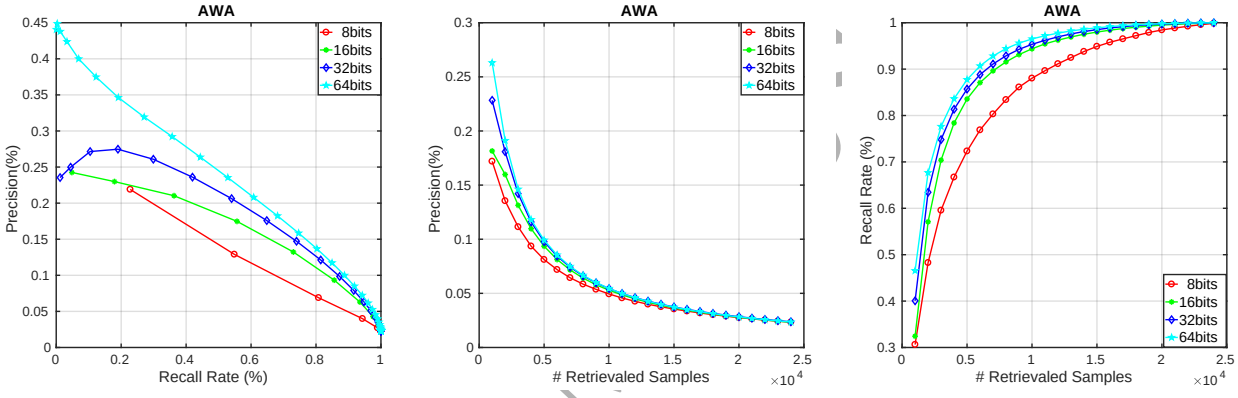
4.4. Detailed Analysis

4.4.1. Precision and Recall

Since AWA has 50 categories and CIFAR-10 has 10 classes, which can be selected as two representative datasets, represent large and small class scale dataset respectively, we draw three types of curves, Precision-Recall (P-R), Precision@N retrieved images, Recall@N retrieved images on both datasets AWA and CIFAR-10 in figure (7) and figure (8). In figure (7), it can be observed that the performance is getting better when the code length is increasing, while in figure (8), these curves are not separate clearly, meaning that the the performance on different code length is nearly same as each other. Therefore, these phenomenons also support the fact that when we have longer code

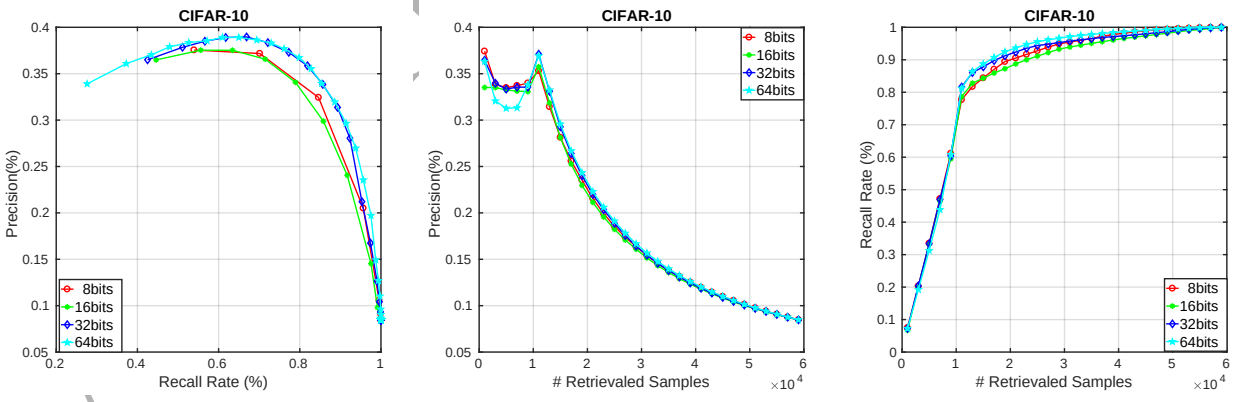
Methods	Query	Retrieved Images	mAP
SitNet (Guo et al., 2017)			0.3383
ITQ (Gong et al., 2013)			0.2066
IMH (Shen et al., 2013)			0.2559
KSH (Liu et al., 2012)			0.1916
DSH (Liu et al., 2016)			0.1917
DHN (Zhu et al., 2016)			0.1666
DNNH (Lai et al., 2015)			0.2702
TSK (Yang et al., 2016)			0.2773
Ours(D)			**0.4220**

Fig. 6. Qualitative results on CIFAR-10 Dataset with queried images of truck. Returned samples with red boxes are false-positive. The mAP is computed with the 20 returned images, from which we can find that our method can outperform all the listed methods.



(a) Precision-Recall curves (b) Precision@N curves (c) Recall@N curves

Fig. 7. Illustration of Precision and Recall Rate for different code length on dataset AWA.



(a) Precision-Recall curves (b) Precision@N curves (c) Recall@N curves

Fig. 8. Illustration of Precision and Recall Rate for different code length on dataset CIFAR-10.

length we can place more semantic binary vectors in orthogonal space, but when the code length is long enough for placing the total class vectors, the performance will not change any more.

The P-R curves which plot the precision and recall rates at different searching Hamming radius $r = \{0, 1, \cdots, \ell - 1, \ell\}$, where ℓ is the code length. In figure (7a) and figure (8a), the first marker of the P-R curves stand for the precision and recall rate at Hamming radius $r = 0$; the second maker means

Table 1. Top 5 shortest average Hamming Distance (aHD) on 32 bits binary codes of retrieved classes on dataset AWA.

query class	Top-5 returned classes									
	1		2		3		4		5	
	HD	class	HD	class	HD	class	HD	class	HD	class
7	10.5	**7**	10.7	21	11.3	49	12.8	37	14.5	28
9	6.6	**9**	6.6	18	7.4	3	8.6	50	15.1	24
23	14.3	21	14.6	**23**	14.6	37	14.6	49	15	16
24	14	**24**	14.2	36	14.5	4	14.5	45	14.9	18
30	11.7	44	12.1	34	12.4	26	12.9	12	13	**30**
31	9.5	**31**	12.7	1	13.1	40	14	15	16.4	16
34	6.9	26	7.6	**34**	7.7	44	10.4	12	10.4	29
41	10.8	**41**	12.1	15	14.3	13	15.5	22	15.6	32
47	13.2	14	13.4	**47**	15.9	24	16.4	28	16.5	19
50	8.3	3	8.6	9	8.7	18	9	**50**	15.5	24

the precision and recall rate at Hamming radius $r = 1$, and so on. It is noticeable that the points at different searching Hamming radius are not aligned, and when we have short Hamming distance, we can obtain high recall rate but low precision.

The Precision@N curves in figure (7b) and figure (8b) represent the precision on N retrieved images, and the Recall@N curves in figure (7c) and figure (8c) stand for the recall rate on N retrieved images. On dataset AWA, both the Precision@N and the Recall@N curves can achieve better performance when the code length is raising. But on dataset CIFAR-10, since the shortest code length is enough for placing all the semantic binary vectors on orthogonal space, we can get similar Precision@N and the Recall@N curves on all the four types of code lengths.

4.4.2. Relationship of classes

In this subsection, we select AWA as the test dataset to find the relationship between the unseen classes and the seen classes. The average Hamming Distances (HD) on 32 bits binary codes of top 5 retrieved classes are recorded in table (1). It can be obviously found that there are 5 classes in the top-1 and 8 classes in top-2 returned classes, only class 50 and class 30 appear in top-4 and top-5 returned classes respectively.

The worst examples are the class 30 and the class 50. The class 30 is *'bat'*, which is not seen in the training classes, and its most semantic similar classes of Word2Vec in the 40 seen classes are *'mouse'*(class 44), *'rat'*(class 34), *'hamster'*(class 26) and *'mole'*(class 12), which all have similar visual appearance. Although the returned class of smallest HD is not the class *'bat'*, the variance of Hamming distances between top-1 and top-5 classes is very small, and the biggest gap between them is only 1.3, which confirms that our method can transfer semantic knowledge of the seen classes to the unseen classes. The class 50 is *'dolphin'*, and its three most similar semantic classes are *'killer+whale'*(class 3), *'blue+whale'*(class 9) and *'humperback+whale'*(class 18), which can also be verified in table (1).

5. Conclusion

In this paper, we propose a new zero-shot hashing method, which projects both images and class names into an orthogonal binary space, where the generated binary codes are orthogonal to each other when they belong to different categories, otherwise to be same. This strategy guarantees that the binary codes of different classes have equal Hamming distance, which can create more discriminative space within limited code length than conventional max-margin loss. We also extend our method with a deep model, which convert the linear projection of hash function into non-linear one. Extensive experiments have been carried out and the results show that our method of both linear model and deep model can get competitive results comparing to the state-of-the-art methods.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Jiangsu Province (SBK2018020020) and the Major Special Project of Core Electronic Devices, High-end Generic Chips and Basic Software (2015ZX01041101).

References

Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C., 2016. Label-embedding for image classification. IEEE transactions on pattern analysis and machine intelligence 38, 1425–1438.

Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B., 2015. Evaluation of output embeddings for fine-grained image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2927–2936.

Al-Halah, Z., Stiefelhagen, R., 2017. Automatic discovery, association estimation and learning of semantic attributes for a thousand categories, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Bakhtiary, A.H., Lapedriza, A., Masip, D., 2017. Winner takes all hashing for speeding up the training of neural networks in large class problems. Pattern Recognition Letters 93.

Bartels, R.H., Stewart, G.W., 1972. Solution of the matrix equation AX + XB = C [F4]. Communications of the ACM 15, 820–826.

Carreira-Perpinán, M.A., Raziperchikolaei, R., 2015. Hashing with binary autoencoders, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 557–566.

Chen, J., Wang, Y., Luo, L., Yu, J.G., Ma, J., 2016. Image retrieval based on image-to-class similarity. Pattern Recognition Letters 83, 379–387.

Demirel, B., Cinbis, R.G., Cinbis, N.I., 2017. Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 248–255.

Do, T.T., Doan, A.D., Cheung, N.M., 2016. Learning to hash with binary deep neural network, in: European Conference on Computer Vision, Springer. pp. 219–234.

Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J., 2015. Deep hashing for compact binary codes learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2475–2483.

Ferrari, V., Zisserman, A., 2008. Learning visual attributes, in: Advances in Neural Information Processing Systems, pp. 433–440.

Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al., 2013. Devise: A deep visual-semantic embedding model, in: Advances in neural information processing systems, pp. 2121–2129.

Fu, Y., Hospedales, T.M., Xiang, T., Fu, Z., Gong, S., 2014. Transductive multi-view embedding for zero-shot recognition and annotation, in: European Conference on Computer Vision, Springer. pp. 584–599.

Gionis, A., Indyk, P., Motwani, R., et al., 1999. Similarity search in high dimensions via hashing, in: Proceedings of the 25th Very Large Database (VLDB) Conference, pp. 518–529.

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

Gui, J., Liu, T., Sun, Z., Tao, D., Tan, T., 2017. Fast supervised discrete hashing. IEEE Transactions on Pattern Analysis and Machine Intelligence .

Guo, Y., Ding, G., Han, J., Gao, Y., 2017. Sitnet: Discrete similarity transfer network for zero-shot hashing, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1767–1773.

Guo, Y., Ding, G., Jin, X., Wang, J., 2016. Transductive zero-shot recognition via shared model space learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 3494–5000.

He, K., Wen, F., Sun, J., 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2938–2945.

Huang, S., Elhoseiny, M., Elgammal, A., Yang, D., 2015. Learning hypergraph-regularized attribute predictors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 409–417.

Kankuekul, P., Kawewong, A., Tangruamsub, S., Hasegawa, O., 2012. Online incremental attribute-based zero-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 3657–3664.

Kodirov, E., Xiang, T., Gong, S., 2017. Semantic autoencoder for zero-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Krizhevsky, A., 2009. Learning multiple layers of features from tiny images. Technical Report. University of Toronto.

Lai, H., Pan, Y., Liu, Y., Yan, S., 2015. Simultaneous feature learning and hash coding with deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3270–3278.

Lampert, C.H., Nickisch, H., Harmeling, S., 2009. Learning to detect unseen object classes by between-class attribute transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 951–958.

Lampert, C.H., Nickisch, H., Harmeling, S., 2014. Attribute-based classification for zero-shot visual object categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence 36, 453–465.

Lin, K., Lu, J., Chen, C.S., Zhou, J., 2016. Learning compact binary descriptors with unsupervised deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1183–1192.

Liu, H., Wang, R., Shan, S., Chen, X., 2016. Deep supervised hashing for fast image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2064–2072.

Liu, L., Shao, L., Shen, F., Yu, M., 2017a. Discretely coding semantic rank orders for supervised image hashing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1425–1434.

Liu, L., Yu, M., Shao, L., 2017b. Latent structure preserving hashing. International Journal of Computer Vision 122, 439–457.

Liu, L., Yu, M., Shao, L., 2017c. Learning short binary codes for large-scale image retrieval. IEEE Transactions on Image Processing 26, 1289–1299.

Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F., 2012. Supervised hashing with kernels, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE. pp. 2074–2081.

Liu, W., Wang, J., Kumar, S., Chang, S.F., 2011. Hashing with graphs, in: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1–8.

Long, T., Xu, X., Shen, F., Liu, L., Xie, N., Yang, Y., 2017a. Zero-shot learning via discriminative representation extraction. Pattern Recogniton Letters doi:https://doi.org/10.1016/j.patrec.2017.09.030.

Long, Y., Liu, L., Shao, L., 2016. Attribute embedding with visual-semantic ambiguity removal for zero-shot learning, in: Proceedings of the British Machine Vision Conference.

Long, Y., Liu, L., Shao, L., Shen, F., Ding, G., Han, J., 2017b. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Lu, J., Li, J., Yan, Z., Zhang, C., 2017. Zero-shot learning by generating pseudo feature representations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Lv, X., Duan, F., Lv, X., Duan, F., 2017. Metric learning via feature weighting for scalable image retrieval. Pattern Recognition Letters doi:https://doi.org/10.1016/j.patrec.2017.09.026.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space, in: International Conference on Learning Representations.

Miller, G.A., 1995. Wordnet: a lexical database for english. Communications of the ACM 38, 39–41.

Morgado, P., Vasconcelos, N., 2017. Semantically consistent regularization for zero-shot recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Shen, F., Shen, C., Liu, W., Tao Shen, H., 2015. Supervised discrete hashing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 37–45.

Shen, F., Shen, C., Shi, Q., Van Den Hengel, A., Tang, Z., 2013. Inductive hashing on manifolds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1562–1569.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

Wu, G., Liu, L., Guo, Y., Ding, G., Han, J., Shen, J., Shao, L., 2017. Unsupervised deep video hashing with balanced rotation, Proceedings of the 26th International Joint Conference on Artificial Intelligence.

Xian, Y., Akata, Z., Sharma, G., Nguyen, Q., Hein, M., Schiele, B., 2016. Latent embeddings for zero-shot classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 69–77.

Yang, H.F., Lin, K., Chen, C.S., 2017. Supervised learning of semantics-preserving hash via deep convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence .

Yang, Y., Luo, Y., Chen, W., Shen, F., Shao, J., Shen, H.T., 2016. Zero-shot hashing via transferring supervised knowledge, in: Proceedings of the 2016 ACM on Multimedia Conference, ACM. pp. 1286–1295.

Yu, F.X., Cao, L., Feris, R.S., Smith, J.R., Chang, S.F., 2013. Designing category-level attributes for discriminative visual recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 771–778.

Zhang, H., Liu, L., Long, Y., Shao, L., 2018. Unsupervised deep hashing with pseudo labels for scalable image retrieval. IEEE Transactions on Image Processing 27, 1626–1638.

Zhang, L., Xiang, T., Gong, S., 2017. Learning a deep embedding model for zero-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Zhu, H., Long, M., Wang, J., Cao, Y., 2016. Deep hashing network for efficient similarity retrieval, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2415–2421.