Deep Supervised Hashing with Nonlinear Projections[*]

Sen Su, Gang Chen, Xiang Cheng, Rong Bi
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
{susen, chg, chengxiang, birong}@bupt.edu.cn

Abstract

Hashing has attracted broad research interests in large scale image retrieval due to its high search speed and efficient storage. Recently, many deep hashing methods have been proposed to perform simultaneous nonlinear feature learning and hash projection learning, which have shown superior performance compared to hand-crafted feature based hashing methods. Nonlinear projection functions have shown their advantages over linear ones due to their powerful generalization capabilities. To improve the performance of deep hashing methods by generalizing projection functions, we propose the idea of implementing a *pure* nonlinear deep hashing network architecture. By consolidating the above idea, this paper presents a Deep Supervised Hashing architecture with Nonlinear Projections (DSHNP). In particular, soft decision trees are adopted as the nonlinear projection functions, since they can generate differentiable nonlinear outputs and can be trained with deep neural networks in an end-to-end way. Moreover, to make the hash codes as independent as possible, we design two regularizers imposed on the parameter matrices of the leaves in the soft decision trees. Extensive evaluations on two benchmark image datasets show that the proposed DSHNP outperforms several state-of-the-art hashing methods.

1 Introduction

With the explosive growth in the volume of images on the Web, large scale image retrieval has become an emerging need and attracted increasing attention. Among the existing techniques for large scale image retrieval, hashing is a popular and effective solution, which enables fast search and efficient storage. The basic idea of hashing methods is to construct a set of hash functions to map images into binary hashes such that similar images are mapped into similar hash codes, then the hamming distance between every two hash codes is used for fast approximate similarity computations and nearest neighbor searches. Most existing hashing methods for image retrieval can be divided into three phases. First, each input image is represented by a vector of feature descriptors. Second, projection functions [Wang *et al.*, 2010b; Kong and Li, 2012] are generated or learned to encode the feature vector into a low-dimensional real-valued vector. Third, the real-valued vector obtained in the second phase is quantized into a binary code. In this paper, we focus on the first and second phases, and adopt the common single bit quantization which is generally used in hashing methods for the third phase.

For the first phase, lots of existing methods utilize hand-crafted descriptors as feature representations of the images. Since the hand-crafted feature construction is independent of the second phase of hashing, the hand-crafted representations may not be tailored to the coding process. Since deep convolutional neural networks (DCNN) have achieved great success in image classification task, many deep hashing methods have been proposed recently to adopt a deep model for feature learning [Xia *et al.*, 2014; Lai *et al.*, 2015; Zhao *et al.*, 2015; Zhang *et al.*, 2015; Li *et al.*, 2016; Yao *et al.*, 2016; Cao *et al.*, 2016]. For the second phase, the hash task needs a projection process which projects the high-dimensional deep feature vector into a low-dimensional approximate hash codes, which is different from the image classification task which directly optimizes the deep features by classification loss (e.g., softmax loss). While linear perceptron functions are widely adopted in existing deep hashing methods, nonlinear projection functions have shown improved performance over linear ones [Salakhutdinov and Hinton, 2009; Liu *et al.*, 2012; Lin *et al.*, 2013; 2014; Shen *et al.*, 2015; Liong *et al.*, 2015]. Thus, we seek to improve the performance of deep hashing methods by generalizing projection functions. In this paper, we investigate how to design a *pure* nonlinear network for hash task to combine nonlinear feature learning models with nonlinear projection functions.

To implement the idea of unifying nonlinear feature learning with nonlinear projection learning, this paper proposes a Deep Supervised Hashing architecture with Nonlinear Projections, which is referred to as DSHNP. In particular, we use deep convolutional neural network to learn feature representations from pixels, and utilize soft decision trees as projection functions which map the learned image representations

[*]This work was supported by National Natural Science Foundation of China under grant 61502047. We acknowledge anonymous reviewers for helpful comments.

into the intermediate real-valued results of hash codes in a nonlinear way. The reasons for choosing soft decision trees lie in that they not only inherit the nonlinear mapping capacity of decision trees, but also make the outputs differentiable such that they can be combined with deep learning architectures whose learning process is performed by back propagation. In addition, to handle the redundancy problem in the hash codes, we design two novel regularizers, i.e., parallel and orthogonal regularizers, which are imposed on the parameter matrices of leaf nodes in soft decision trees. To train DSHNP, we adopt pair-like samples and utilize the negative log-likelihood loss as the object function along with the above two regularizers.

Our main contributions are outlined as follows:

- We propose a *pure* nonlinear deep hashing network architecture (DSHNP) for supervised hashing. In particular, DSHNP uses a deep convolutional neural network for feature learning, utilizes soft decision trees for nonlinear projection, and is trained end-to-end by optimizing a pairwise loss function.
- To reduce the redundancy in the learned hash codes, we propose two novel regularizers, i.e., parallel and orthogonal regularizers, which are imposed on the weighting parameters of leaf nodes in the soft decision trees.
- We conduct extensive experiments on real datasets to evaluate DSHNP, and the experimental results demonstrate the superior performance of DSHNP over state-of-the-art hashing methods.

2 Related Work

Here we mainly discuss the recent works on deep hashing methods. Xia et al. [Xia *et al.*, 2014] extend TSH [Lin *et al.*, 2013] to propose a deep hashing method CNNH. They first learn hash codes from the pairwise labels, then regard projection function learning as a classification problem solved along with feature learning by deep convolutional neural network (DCNN) which is proposed for image classification. Realizing the limitations of CNNH which solves the hashing problem in a two-stage way, Lai et al. [Lai *et al.*, 2015] propose a one-stage supervised hashing method NINH via a deep architecture, and design a divide-and-encode strategy to reduce the redundancy among the hash bits. Zhao et al. [Zhao *et al.*, 2015] combine semantic ranking and deep learning model in the proposed DSRH to address the problem of preserving multilevel semantic similarity between multilabel images. Zhang et al. [Zhang *et al.*, 2015] pose hashing learning as a problem of regularized similarity learning, and propose a supervised bit-scalable deep hashing framework DRSCH with an extra weight matrix learned to calculate the weighted hamming affinity. Li et al. [Li *et al.*, 2016] propose an end-to-end hashing learning framework DPSH which performs simultaneous feature learning and projection function learning for applications with pairwise labels. Cao et al. [Cao *et al.*, 2016] propose a novel Deep Quantization Network (DQN) architecture for supervised hashing, which learns image representations for hash coding and formally controls the quantization error. Liu et al. [Liu *et al.*, 2016] design a novel

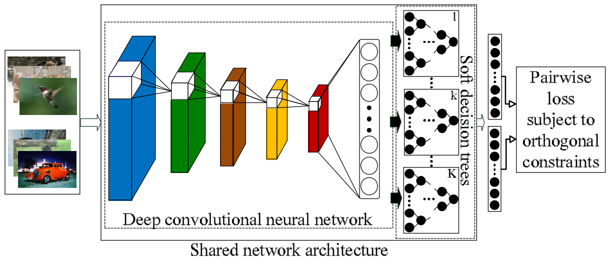
Deep convolutional neural network
Soft decision trees
1
k
K
Pairwise loss subject to orthogonal constraints
Shared network architecture

Figure 1: the architecture of Deep Supervised Hashing with Nonlinear Projections (DSHNP). The input to DSHNP is in the form of image pairs. A shared DCNN is implemented for learning image feature representations. Shared soft decision trees are utilized as the projection layer, which nonlinearly maps the constructed features into dimensional-reduced projections. A pairwise loss with orthogonality constraints is minimized to get the optimal hash function.

loss function to preserve the pairwise similarity and reduce the discrepancy between the real-valued outputs and the desired discrete values. Yao et al. [Yao *et al.*, 2016] propose the idea of co-training for hashing, by jointly learning projections from image representations to hash codes and classification, and present a novel deep semantic-preserving and ranking-based hashing (DSPRH) architecture. However, the above feature learning based deep hashing methods directly impose a fully-connected layer as the projection function which linearly maps the learned features to approximate hash codes. The proposed *pure* nonlinear deep hashing network architecture DSHNP in this paper advances the above deep hashing methods in the coupling of nonlinear feature learning and nonlinear projections.

Recently, quite a few works have paid attention to a differentiable use of tree structures in deep learning, which are related to our DSHNP. Irsoy et al. [Irsoy and Alpaydin, 2014] implement autoencoder by soft decision trees instead of multilayer perceptron. In the DNDF model proposed by Kontschieder et al. [Kontschieder *et al.*, 2015], decision trees play the role of classifier in DCNN instead of the softmax layer. Lee et al. [Lee *et al.*, 2015] redesign the pooling layer of convolutional neural network by learning a pooling function in the form of a tree-structured fusion of pooling filters. Thus, these works are different from our DSHNP which adopts decision trees as the nonlinear projection part of hashing.

3 Deep Supervised Hashing with Nonlinear Projections

In this section, we will present the proposed DSHNP in details. Figure 1 illustrates an overview of DSHNP, which consists of three components: (1) a shared DCNN for learning image representations; (2) shared soft decision trees to nonlinearly map image representations into dimension-reduced projections; (3) a pairwise loss subject to orthogonality constraints. In what follows, we will give a description of model formulation, then discuss the details of soft decision trees and orthogonality constraints, at last present the loss function and its relaxation.

3.1 Formulation

Suppose there is an image i whose raw image data is denoted by x_i. The feature representation vector extracted by the shared DCNN is denoted as $y_i = \phi(x_i; \Theta)$, where Θ is the set of all parameters in the DCNN architecture, and $\phi(x_i; \Theta)$ denotes the output of the shared DCNN associated with image i. The projection layer is used to map the feature representation vector y_i into the low-dimensional real-valued projections $\boldsymbol{F}(y_i; W)$, where $\boldsymbol{F}(\cdot)$ is the set of the projection functions, W is the set of parameters in the projection layer and $\mathcal{F}_i$ is denoted as the hash projections of image i for notation brevity. Specifically, the projection layer is consisted of K soft decision trees, where K is the length of hash codes. Finally the real-valued projections are converted into a binary code by quantization. In this paper, to calculate the hamming distance conveniently, the binary digits are set as $\{-1, 1\}$ instead of $\{0, 1\}$. In particular, the hash code h_i of image i is computed by:

$$h_i = sgn(\boldsymbol{F}(\phi(x_i; \Theta); W)), \tag{1}$$

where $sgn(\cdot)$ is the element-wise sign function which returns 1 if the input is positive and returns -1 otherwise. The hamming distance d_{ij} between the hash codes of images i and j is calculated as:

$$d_{ij} = \frac{1}{2}(K - h_i^T h_j). \tag{2}$$

3.2 Nonlinear Projection Functions

Decision trees which are efficient models for classification and regression [Bosch *et al.*, 2007; Criminisi and Shotton, 2013] have played a central role in artificial intelligence for applications that require "discrete" reasoning, and are often intuitively appealing. Thus, it is intuitively suitable to adopt decision trees as hash projections which are preferred to directly map the high-dimensional feature vectors into discrete hamming spaces. Moreover, since decision trees can easily deal with a large number of data [Caruana *et al.*, 2008] and have the nonlinear mapping capacity, we choose decision trees as the nonlinear projection functions. Furthermore, to unify decision trees with deep learning architecture in this paper, we adopt the soft decision trees [Irsoy *et al.*, 2012] instead of the hard ones. As opposed to the hard internal node which makes a hard choice to redirect samples to one of its children, a soft internal node redirects samples to both of its children with probabilities calculated by a sigmoid gating function $\sigma(x) = \frac{1}{1+e^{-x}}$. For the left child, the probability is $\sigma(x)$; for the right one, it is $1 - \sigma(x)$. In fact, the hard decision tree is a special case where $g(x) \in \{0, 1\}$. As the outputs of soft decision trees are differentiable, we can train a deep neural network combined with soft decision trees via backpropagation in an end-to-end way.

In this paper, we leverage the soft decision trees as nonlinear projection functions in the following manner. We denote the mth node in the lth level of the kth soft decision tree (i.e., the node at the position (l, m) in the kth tree) as node $(l, m)k$. The root node of the kth tree is $(1, 1)k$. Each tree node is multivariate, which takes the feature representation vector y_i of image i as input instead of a single feature. The output of node $(l, m)k$ is recursively calculated as the weighted average of the outputs of its left and right children:

$$f_{(l,m)k}(y_i) = \begin{cases} W_{(l,m)k}^T y_i, & if\ (l,m)k\ is\ leaf\ node; \\ \\ \sigma(W_{(l,m)k}^T y_i) f_{(l,m)k,L}(y_i) \\ \ + (1 - \sigma(W_{(l,m)k}^T y_i)) f_{(l,m)k,R}(y_i), & otherwise, \end{cases} \tag{3}$$

where $W_{(l,m)k}$ is the parameter vector of node $(l, m)k$. Bias $b_{(l,m)k}$ is included in $W_{(l,m)k}$ with a corresponding fixed input of 1 for simplicity. The parameter vectors at the position (l, m) of all K soft decision trees can compose a parameter matrix $W_{(l,m)}$, then $W_{(l,m)k}$ can be regarded as the kth column vector of $W_{(l,m)}$. The root node is the terminal node to produce the overall output of the tree. The output of each soft decision tree is considered as a single real-valued hash bit. All K soft decision trees are of the same structure. In particular, the level of the leaf node is pre-specified to be ll rather than "grown" as in the traditional decision trees. When ll is set to 1, the projection layer made of soft decision trees is equivalent to a fully-connected layer (i.e., linear projection function). The output of the projection layer for image i is calculated as follows:

$$\begin{aligned} \mathcal{F}_i &= f_{(1,1)}(y_i) \\ &= \sigma(W_{(1,1)}^T y_i) \circ ... \circ W_{(ll,1)}^T y_i + ... \\ &+ (1 - \sigma(W_{(1,1)}^T y_i)) \circ ... \circ W_{(ll,2^{ll-1})}^T y_i \end{aligned} \quad , \tag{4}$$

where $\circ$ is the Hadamard product (i.e., element-wise product).

3.3 Orthogonality Constraints

One way to reduce redundancy in hash codes is to make the projection directions orthogonal [Wang *et al.*, 2010a; Tang *et al.*, 2015]. Since $W_{(ll,m)}^T y_i$ in Eq.(4) can be considered as linear projections of the feature representation y_i, the projection output $\mathcal{F}_i$ can be regarded as the weighted sum of linear projections of y_i. Note that the routing weights have nothing to do with the projection directions. Therefore, to make the projection directions orthogonal, each linear projection (i.e., the leaf parameter matrix $W_{(ll,m)}$) should satisfy the following two constraints: (1) all the parameter vectors of the leaf nodes belonging to the same soft decision tree are parallel; (2) the parameter vectors of the leaf nodes belonging to every two soft decision trees are orthogonal. In this way, we can make the whole projection directions orthogonal, and achieve the goal of reducing redundancy in the learned hash codes.

To satisfy the parallel constraint, we can impose the following constraint on the parameter vectors of every two leaf nodes at different positions (e.g., (ll, m), (ll, m')) belonging to the same soft decision tree (e.g., the kth soft decision tree):

$$\left| \frac{W_{(ll,m)k}^T W_{(ll,m')k}}{\left\| W_{(ll,m)k} \right\| \cdot \left\| W_{(ll,m')k} \right\|} \right| = 1. \tag{5}$$

In addition, to satisfy the orthogonal constraint, as the parameter vectors at the position (ll, m) of all K decision trees constitute the leaf parameter matrix $W_{(ll,m)}$, we just need to guarantee this leaf parameter matrix is orthogonal. Thus, we can impose the following constraint on this leaf parameter matrix:

$$W^T_{(ll,m)} W_{(ll,m)} = I. \tag{6}$$

Under the parallel constraints imposed on every two leaf parameter vectors at different positions belonging to the same tree, we do not need to consider the orthogonal constraints between the leaf parameter matrices at different positions.

Proposition 1 (projection orthogonality condition). If the leaf parameter matrices in the soft decision trees which are used as projection functions satisfy (5) and (6), then the projection directions are orthogonal.

Proof. Due to the parallel constraint (5), all leaf parameter vectors of the kth soft decision tree can be linearly-represented by a single vector $\boldsymbol{\beta}_k$. The parameter vector $W_{(ll,m)k}$ can be calculated as follows:

$$W_{(ll,m)k} = a_{(ll,m)k} \boldsymbol{\beta}_k, \tag{7}$$

where $a_{(ll,m)k}$ denotes the weight of the leaf parameter vector $W_{(ll,m)k}$ relative to $\boldsymbol{\beta}_k$, which is a scalar value. Due to the orthogonal constraint (6), the K vectors $[\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_k, ..., \boldsymbol{\beta}_K]$ are orthogonal. The nonlinear weighting coefficients by which the linear projections are multiplied in Eq.(4) can be regarded as a K-dimensional vector associated with y which is denoted by $\boldsymbol{\Gamma}_{(ll,m)}(y)$. For example, when the level of the leaf node is 2, the nonlinear weighting coefficients of the leaf parameter matrix $W_{(2,2)}$, i.e., $1 - \sigma(W^T_{(1,1)} y)$, can be denoted by $\boldsymbol{\Gamma}_{(2,2)}(y)$.

For any image i, the projection output $\mathcal{F}_{ik}$ of its kth hash bit h_{ik} can be defined as:

$$\begin{aligned} \mathcal{F}_{ik} &= \sum_{m=1}^{2^{ll-1}} \boldsymbol{\Gamma}_{(ll,m)k}(y_i) \cdot W^T_{(ll,m)k} y_i \\ &= (\sum_{m=1}^{2^{ll-1}} \boldsymbol{\Gamma}_{(ll,m)k}(y_i) a_{(ll,m)k}) \cdot {\boldsymbol{\beta}_k}^T y_i. \end{aligned} \tag{8}$$

$\mathcal{F}_{ik}$ in the above equation can be regarded as the product of the linear projection of image feature representation y_i at the direction of $\boldsymbol{\beta}_k$ and the sum of the corresponding nonlinear weighting coefficients. Since the vectors $[\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_k, ..., \boldsymbol{\beta}_K]$ are orthogonal to each other, the ultimate projection directions are orthogonal based on these two constraints.

3.4 Loss Function

As proved in [Li *et al.*, 2016; Cao *et al.*, 2016], taking the pairwise loss as the training objective can bring higher accuracy and be optimized more easily than the triplet loss. Thus, we adopt the pairwise negative log-likelihood loss proposed in [Li *et al.*, 2016] to train our deep hashing model.

A probability P_{ij} that image i should be similar to image j is defined via a sigmoid function based on the hamming distance between the hash codes of these two images. In particular, the larger the hamming distance is, the smaller the similar probability is. Due to Eq.(2), since K is a preset value, we use $-\frac{1}{2} h_i^T h_j$ to represent the difference of the hash codes for simplicity. Then the probability P_{ij} is defined as follows:

$$P_{ij} = \sigma(-1 \cdot (-\frac{1}{2} h_i^T h_j)) = \frac{1}{1 + e^{-\frac{1}{2} h_i^T h_j}}. \tag{9}$$

The cross entropy cost function is applied which penalizes the deviation of the model output probabilities from the desired probabilities:

$$\begin{aligned} C_{ij} &= -S_{ij} \log P_{ij} - (1 - S_{ij}) \log(1 - P_{ij}) \\ &= -S_{ij}(\frac{1}{2} h_i^T h_j) + \log(1 + e^{\frac{1}{2} h_i^T h_j}), \end{aligned} \tag{10}$$

where $S_{ij} \in \{0, 1\}$ is the known desired probability that i and j should be similar, i.e., $S_{ij} = 1$ if i and j are similar otherwise $S_{ij} = 0$. By appending our proposed orthogonality constraints to the cross entropy cost, we can formulate our hash learning problem as follows:

$$\begin{aligned} &\min_{\Theta, W} \mathcal{L} = \sum_{S_{ij} \in \{0,1\}} (-S_{ij}(\tfrac{1}{2} h_i^T h_j) + \log(1 + e^{\frac{1}{2} h_i^T h_j})). \\ &s.t. \quad h_i \in \{-1, 1\}^K, \forall i = 1, 2, ..., N \\ &\qquad W \; satisfies \; (5)(6) \end{aligned} \tag{11}$$

3.5 Relaxation

To address the optimization problem caused by the discontinuous hamming distance, we relax it by taking a continuous form $-\frac{1}{2} h_i^T h_j = -\frac{1}{2} \mathcal{F}_i^T \mathcal{F}_j$. Then, the relaxed cross entropy can be rewritten as:

$$\tilde{C}_{ij} = -S_{ij} \cdot (\frac{1}{2} \mathcal{F}_i^T \mathcal{F}_j) + log(1 + e^{\frac{1}{2} \mathcal{F}_i^T \mathcal{F}_j}). \tag{12}$$

To optimize the problem in Eq.(11), we move the parallel and orthogonal constraints to the following regularization terms. One is the relaxation of the parallel constraint to make all leaf parameter vectors belonging to the same soft decision tree parallel, which is defined as follows:

$$R^{(1)}_{1,(ll,m)k} = (W^T_{(ll,m)k} W_{(ll,m+1)k})^2, \tag{13}$$

$$R^{(2)}_{1,(ll,m)k} = \left\| W_{(ll,m)k} \right\|^2 \cdot \left\| W_{(ll,m+1)k} \right\|^2, \tag{14}$$

$$\begin{aligned} R_1 &= \sum_{k=1}^{K} \sum_{m=1}^{2^{ll-1}} (R^{(1)}_{1,(ll,m)k} - R^{(2)}_{1,(ll,m)k}). \\ &(when \; m = 2^{ll-1}, m + 1 = 1) \end{aligned} \tag{15}$$

The other is the relaxation of the orthogonal constraint to keep the leaf parameter vectors at the same position of all the soft decision trees orthogonal, which is defined as follows:

$$R_2 = \sum_{m=1}^{2^{ll-1}} \left\| W^T_{(ll,m)} W_{(ll,m)} - I \right\|_F^2, \tag{16}$$

where $\|\cdot\|_F$ is the Frobenius norm on matrices.

Thus, the relaxed overall loss function can be rewritten as:

$$\tilde{\mathcal{L}} = \sum_{S_{ij} \in \{0,1\}} \tilde{C}_{ij} + \frac{\lambda}{2}(R_1 + R_2). \tag{17}$$

Given the above objective function Eq.(17), the hashing model can be trained in a batch-process fashion by mini-batch gradient descent. The model parameters Θ and W can be optimized via back-propagation algorithm with the computed derivatives $\frac{\partial \tilde{\mathcal{L}}}{\partial \mathcal{F}_i}$ and $\frac{\partial \tilde{\mathcal{L}}}{\partial W}$ (the computations are straightforward and then omitted due to the space limit).

4 Experiments

4.1 Datasets and Evaluation Metric

In our experiments, we utilize two benchmark image datasets, CIFAR-10 and FLICKR. Following [Liu *et al.*, 2012; Lai *et al.*, 2015; Li *et al.*, 2016], we set up our experiments as the settings described below.

CIFAR-10

The CIFAR-10 dataset [Krizhevsky, 2009] is a single-label dataset which consists of 60,000 32 $\times$ 32 colour images in 10 classes, with 6,000 images per class. For training, we randomly select 5,000 images with 500 images per class; for testing, we randomly select 1,000 images with 100 images per class. Two images will be considered to be similar if they share the same class label. For hashing methods which use hand-crafted features, a 512-D GIST vector is used to represent each image in CIFAR-10. For deep hashing methods, the raw image pixels are directly utilized as the input.

FLICKR

The FLICKR dataset [Huiskes and Lew, 2008] consists of 25,000 images collected from Flickr, where each image is labeled with several labels from the 38 semantic concepts. Two images are regarded to be similar if they share at least one common concept and vice versa. We randomly select 1,000 images as the test query set, and 4,000 images as the training set. Following [Srivastava and Salakhutdinov, 2014], we use a 3,857-D low-level features extracted from images as the hand-crafted features to represent these images for hand-crafted feature based hashing methods. We utilize the raw pixels of these images as the input of deep hashing methods.

Similar to most existing studies, we utilize the mean average precision (MAP) to measure the accuracy of DSHNP and other baselines.

4.2 Implementation Details

We implement the DSHNP model based on the open-source MatConvNet framework [Vedaldi and Lenc, 2015]. The experiments are carried out on a workstation with Intel Xeon E5-2620, 64GB RAM, and NVIDIA Titan X with CUDA-7.5 and cuDnn v5.0. We utilize GoogLeNet [Szegedy *et al.*, 2015] as our basic DCNN architecture and remove its last linear and softmax layers. We fine-tune the pre-trained GoogLeNet model[1], and train soft decision trees, all via back-propagation. To handle the over-fitting problem in the soft decision tree layer, we add a batch normalization layer after each decision tree node. When optimizing the parameters of DSHNP, we use the mini-batch stochastic gradient descent with 0.9 momentum, $5e^{-4}$ weight decay and 128 mini-batch size. The parameter settings of tree level ll and regularization weight λ will be discussed in the section 4.3.

4.3 Empirical Analysis

Effect of the Nonlinear Projections

In this part, we validate the effectiveness of our proposed nonlinear projection part (i.e., soft decision trees) in DSHNP.

[1]http://www.vlfeat.org/matconvnet/models/imagenet-googlenet-dag.mat

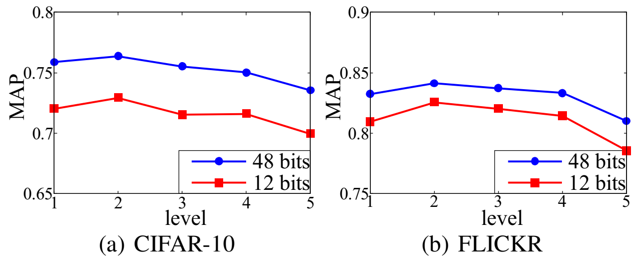

Figure 2: Effect of the Nonlinear Projections

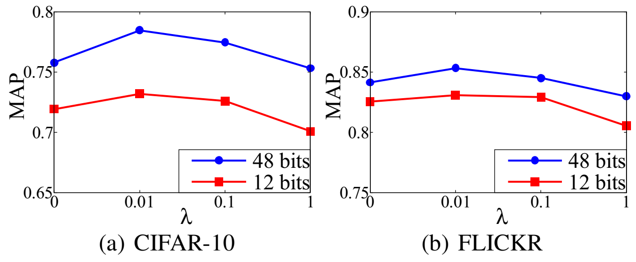

Figure 3: Effect of the Orthogonality Regularizers

Without loss of generality, we only test the cases when the code length $K = 12$ and 48 in our DSHNP without the orthogonality regularizers. We evaluate the performance of DSHNP by varying tree level ll from 1 to 5 (note that when $ll = 1$ DSHNP becomes equivalent to a deep hashing model with linear projections). Figure 2 shows the experimental results of DSHNP with different tree levels. From the results, we can find that: first, DSHNP with $ll = 1$ performs inferior to that with $ll = 2$, which shows the performance boost provided by nonlinear projections and demonstrates the advantage of our *pure* nonlinear deep hashing network; second, when ll becomes larger than 2, the performance of DSHNP gets worse, the reason may lie in that the larger amount of parameters in a deeper soft decision tree makes the model easier to become over-fitting and harder to be optimized. Based on the above observations, we empirically set the tree level $ll = 2$ in the following experiments.

Effect of the Orthogonality Regularizers

In this part, we validate the effectiveness of the proposed orthogonality regularizers. Similar to the above experiments of nonlinear projections, we test the cases when the code length $K = 12$ and 48 in our DSHNP with tree level $ll = 2$. We test these models without the regulariers (i.e., the weighting parameter $\lambda = 0$) and with $\lambda = 0.01, 0.1, 1$. From the comparison results in Figure 3, we can make the following observations: first, without the orthogonality regularizers ($\lambda = 0$), DSHNP gets an inferior performance to those with the regularizers, suggesting that our orthogonality regularizers which aims to reduce the bit redundancy is advantageous; second, the MAP performances can be improved when setting λ under a reasonable range (e.g. $[0.01, 0.1]$). Based on the above observations, we empirically set $\lambda = 0.01$ in the following experiments.

Table 1: Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on CIFAR-10 and FLICKR

Method	CIFAR-10				FLICKR			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
DSHNP	**0.731**	**0.758**	**0.766**	**0.784**	**0.831**	**0.839**	**0.848**	**0.853**
DSHNP-1	0.720	0.729	0.745	0.759	0.809	0.820	0.824	0.832
DPSH	0.713	0.727	0.744	0.757	0.791	0.800	0.817	0.828
NINH	0.552	0.566	0.558	0.581	0.783	0.789	0.791	0.802
CNNH	0.465	0.521	0.521	0.532	0.749	0.761	0.768	0.776
FastH+DCNN	0.630	0.685	0.705	0.721	0.769	0.799	0.805	0.812
FastH	0.387	0.443	0.463	0.483	0.779	0.782	0.785	0.801
SDH	0.316	0.426	0.434	0.444	0.688	0.691	0.704	0.719
KSH	0.362	0.398	0.413	0.426	0.707	0.711	0.723	0.721
ITQ	0.222	0.233	0.235	0.244	0.560	0.563	0.564	0.567
SH	0.190	0.187	0.184	0.189	0.563	0.567	0.571	0.579

4.4 Comparison with the State-of-the-art

Baseline Methods

To evaluate the effectiveness of DSHNP, we compare it against several state-of-the-art hashing methods as follows:

- Unsupervised hashing methods with hand-crafted features: SH [Weiss *et al.*, 2008], ITQ [Gong and Lazebnik, 2011];
- Supervised hashing methods with hand-crafted features and nonlinear projections: KSH [Liu *et al.*, 2012], FastH [Lin *et al.*, 2014], SDH [Shen *et al.*, 2015];
- Supervised hashing methods with DCNN features and nonlinear projections: FastH+DCNN which denotes the best shallow baseline FastH using DCNN features extracted from the pre-trained GoogLeNet model;
- Deep hashing methods with raw image inputs and linear projections: CNNH [Xia *et al.*, 2014], NINH [Lai *et al.*, 2015], DPSH [Li *et al.*, 2016].

There are some other deep hashing methods we do not compare with, such as (1) DQN [Cao *et al.*, 2016], DSH [Liu *et al.*, 2016] and (2) DSPRH [Yao *et al.*, 2016]. The reasons are described as follows: (1) DQN, DSH: they focus on the quantization part instead of the projection part, which are orthogonal to our work and can be used to improve our work. (2) DSPRH: it proposes the idea of co-training between hashing and preserving semantic structures, which is also orthogonal to our work and can be adopted to improve our work.

Results

The experiments evaluate the performance of different hashing methods by varying the number of hashing bits in the range of $\{12, 24, 32, 48\}$. All of the MAP results on both datasets are reported in Table 1. We can find that our DSHNP achieves better performance than the competitors, including unsupervised methods, supervised methods with hand-crafted features and nonlinear projections, and deep hashing methods with feature learning and linear projections. Specifically, DSHNP outperforms the hand-crafted feature based methods with nonlinear projections (FastH, SDH, KSH), which is mainly achieved by fusing feature learning and nonlinear projections. In particular, compared to the best one of the baselines which uses traditional hand-crafted visual features and nonlinear projection functions, FastH, DSHNP outperforms it by very large margins of 31.6% and 5.6% in average MAP on the CIFAR-10 and FLICKR datasets respectively. Comparing with the two-stage deep hashing method CNNH, the superior performance of DSHNP shows the effectiveness of the end-to-end learning framework. To further verify the importance of simultaneous feature learning and projection learning, we report the results of FastH+DCNN, the best hand-crafted feature based method FastH trained with DCNN features. The results show that FastH+DCNN achieves improved performance than FastH, but it is still inferior to DSHNP, which further demonstrates the advantage of the end-to-end learning scheme. Comparing with the deep hashing methods (DPSH, NINH) which simultaneously learn both features and hash codes but utilize linear projections, DSHNP shows superior performance gains in MAP on both datasets, which shows the effectiveness of unifying feature learning with nonlinear projections for hashing. Specifically, compared to the state-of-the-art deep hashing method DPSH, DSHNP achieves absolute increases of 2.4% and 3.4% in average MAP on CIFAR-10 and FLICKR respectively. We futher report the results of DSHNP with $ll = 1$ and $\lambda = 0$ (denoted by DSHNP-1), which can be seen as DPSH adopting GoogLeNet instead of VGG-F [Chatfield *et al.*, 2014]. Once again, the experimental results demonstrate the effectiveness of nonlinear projections.

5 Conclusion

In this paper, we propose a *pure* nonlinear deep hashing network architecture DSHNP, which fuses feature learning functionality and nonlinear projection capacity in an end-to-end learning framework. Particularly, we implement a shared DCNN to learn feature representations of images and leverage soft decision trees to map the learned features into low-dimensional vectors in a nonlinear way. To train this *pure* nonlinear deep hashing model including DCNN and soft decision trees, we minimize a pairwise loss subject to orthogonal constraints which aim to make the generated hash codes as independent as possible. Extensive experiments on real-world image datasets show that our DSHNP achieves better performance than state-of-the-art hashing methods, which demonstrates the advantage of unifying feature learning with nonlinear projections for hashing.

References

[Bosch *et al.*, 2007] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *ICCV*, 2007.

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

[Caruana *et al.*, 2008] Rich Caruana, Nikolaos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML*, 2008.

[Chatfield *et al.*, 2014] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[Criminisi and Shotton, 2013] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

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

[Huiskes and Lew, 2008] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08*, 2008.

[Irsoy and Alpaydin, 2014] Ozan Irsoy and Ethem Alpaydin. Autoencoder trees. *CoRR*, abs/1409.7461, 2014.

[Irsoy *et al.*, 2012] Ozan Irsoy, Olcay Taner Yildiz, and Ethem Alpaydin. Soft decision trees. In *ICPR*, 2012.

[Kong and Li, 2012] Weihao Kong and Wu-Jun Li. Double-bit quantization for hashing. In *AAAI*, 2012.

[Kontschieder *et al.*, 2015] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *ICCV*, 2015.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

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

[Lee *et al.*, 2015] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *CoRR*, abs/1509.08985, 2015.

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

[Lin *et al.*, 2013] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, 2013.

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

[Liong *et al.*, 2015] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2015.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, 2012.

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

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

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

[Srivastava and Salakhutdinov, 2014] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. *JMLR*, 15:2949–2980, 2014.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[Tang *et al.*, 2015] Jinhui Tang, Zechao Li, Meng Wang, and Ruizhen Zhao. Neighborhood discriminant hashing for large-scale image retrieval. *TIP*, 24(9):2827–2840, 2015.

[Vedaldi and Lenc, 2015] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for MATLAB. In *MM*, 2015.

[Wang *et al.*, 2010a] Jun Wang, Ondrej Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[Wang *et al.*, 2010b] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *ICML*, 2010.

[Weiss *et al.*, 2008] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, 2008.

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

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

[Zhang *et al.*, 2015] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *TIP*, 24(12):4766–4779, 2015.

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