HashNet: Deep Learning to Hash by Continuation*

Zhangjie Cao[†], Mingsheng Long[†], Jianmin Wang[†], and Philip S. Yu[†‡]
[†]KLiss, MOE; NEL-BDS; TNList; School of Software, Tsinghua University, China
[‡]University of Illinois at Chicago, IL, USA
caozhangjie14@gmail.com {mingsheng,jimwang}@tsinghua.edu.cn psyu@uic.edu

Abstract

Learning to hash has been widely applied to approximate nearest neighbor search for large-scale multimedia retrieval, due to its computation efficiency and retrieval quality. Deep learning to hash, which improves retrieval quality by end-to-end representation learning and hash encoding, has received increasing attention recently. Subject to the ill-posed gradient difficulty in the optimization with sign activations, existing deep learning to hash methods need to first learn continuous representations and then generate binary hash codes in a separated binarization step, which suffer from substantial loss of retrieval quality. This work presents HashNet, a novel deep architecture for deep learning to hash by continuation method with convergence guarantees, which learns exactly binary hash codes from imbalanced similarity data. The key idea is to attack the ill-posed gradient problem in optimizing deep networks with non-smooth binary activations by continuation method, in which we begin from learning an easier network with smoothed activation function and let it evolve during the training, until it eventually goes back to being the original, difficult to optimize, deep network with the sign activation function. Comprehensive empirical evidence shows that HashNet can generate exactly binary hash codes and yield state-of-the-art multimedia retrieval performance on standard benchmarks.

1. Introduction

In the big data era, large-scale and high-dimensional media data has been pervasive in search engines and social networks. To guarantee retrieval quality and computation efficiency, approximate nearest neighbors (ANN) search has attracted increasing attention. Parallel to the traditional indexing methods [21], another advantageous solution is hashing methods [38], which transform high-dimensional media data into compact binary codes and generate similar binary codes for similar data items. In this paper, we will focus on learning to hash methods [38] that build data-dependent hash encoding schemes for efficient image retrieval, which have shown better performance than data-independent hashing methods, e.g. Locality-Sensitive Hashing (LSH) [10].

Many learning to hash methods have been proposed to enable efficient ANN search by Hamming ranking of compact binary hash codes [19, 12, 30, 9, 25, 37, 27, 11, 41, 42]. Recently, deep learning to hash methods [40, 20, 34, 8, 44, 22, 24] have shown that end-to-end learning of feature representation and hash coding can be more effective using deep neural networks [18, 2], which can naturally encode any nonlinear hash functions. These deep learning to hash methods have shown state-of-the-art performance on many benchmarks. In particular, it proves crucial to jointly learn similarity-preserving representations and control quantization error of binarizing continuous representations to binary codes [44, 22, 43, 24]. However, a key disadvantage of these deep learning to hash methods is that they need to first learn continuous deep representations, which are binarized into hash codes in a separated post-step of sign thresholding. By *continuous relaxation*, i.e. solving the discrete optimization of hash codes with continuous optimization, all these methods essentially solve an optimization problem that deviates significantly from the hashing objective as they cannot learn *exactly binary* hash codes in their optimization procedure. Hence, existing deep hashing methods may fail to generate compact binary hash codes for efficient similarity retrieval.

There are two key challenges to enabling deep learning to hash truly end-to-end. First, converting deep representations, which are *continuous* in nature, to *exactly binary* hash codes, we need to adopt the *sign* function $h = \operatorname{sgn}(z)$ as activation function when generating binary hash codes using similarity-preserving learning in deep neural networks. However, the gradient of the sign function is zero for all nonzero inputs, which will make standard back-propagation infeasible. This is known as the *ill-posed gradient* problem, which is the key difficulty in training deep neural networks via back-propagation [14]. Second, the similarity information is usually very sparse in real retrieval systems, i.e., the number of similar pairs is much smaller than the number of dissimilar pairs. This will result in the *data imbalance*

[*] Corresponding author: M. Long (mingsheng@tsinghua.edu.cn).

problem, making similarity-preserving learning ineffective. Optimizing deep networks with *sign* activation remains an open problem and a key challenge for deep learning to hash.

This work presents **HashNet**, a new architecture for deep learning to hash by continuation with convergence guarantees, which addresses the ill-posed gradient and data imbalance problems in an end-to-end framework of deep feature learning and binary hash encoding. Specifically, we attack the *ill-posed gradient* problem in the non-convex optimization of the deep networks with non-smooth sign activation by the *continuation* methods [1], which address a complex optimization problem by smoothing the original function, turning it into a different problem that is easier to optimize. By gradually reducing the amount of smoothing during the training, it results in a sequence of optimization problems converging to the original optimization problem. A novel weighted pairwise cross-entropy loss function is designed for similarity-preserving learning from imbalanced similarity relationships. Comprehensive experiments testify that HashNet can generate exactly binary hash codes and yield state-of-the-art retrieval performance on standard datasets.

2. Related Work

Existing learning to hash methods can be organized into two categories: unsupervised hashing and supervised hashing. We refer readers to [38] for a comprehensive survey.

Unsupervised hashing methods learn hash functions that encode data points to binary codes by training from unlabeled data. Typical learning criteria include reconstruction error minimization [33, 12, 16] and graph learning[39, 26]. While unsupervised methods are more general and can be trained without semantic labels or relevance information, they are subject to the semantic gap dilemma [35] that high-level semantic description of an object differs from low-level feature descriptors. Supervised methods can incorporate semantic labels or relevance information to mitigate the semantic gap and improve the hashing quality significantly. Typical supervised methods include Binary Reconstruction Embedding (BRE) [19], Minimal Loss Hashing (MLH) [30] and Hamming Distance Metric Learning [31]. Supervised Hashing with Kernels (KSH) [25] generates hash codes by minimizing the Hamming distances across similar pairs and maximizing the Hamming distances across dissimilar pairs.

As deep convolutional neural network (CNN) [18, 13] yield breakthrough performance on many computer vision tasks, deep learning to hash has attracted attention recently. CNNH [40] adopts a two-stage strategy in which the first stage learns hash codes and the second stage learns a deep network to map input images to the hash codes. DNNH [20] improved the two-stage CNNH with a simultaneous feature learning and hash coding pipeline such that representations and hash codes can be optimized in a joint learning process. DHN [44] further improves DNNH by a cross-entropy loss and a quantization loss which preserve the pairwise similarity and control the quantization error simultaneously. DHN obtains state-of-the-art performance on several benchmarks.

However, existing deep learning to hash methods only learn continuous codes $\boldsymbol{g}$ and need a binarization post-step to generate binary codes $\boldsymbol{h}$. By continuous relaxation, these methods essentially solve an optimization problem $L(\boldsymbol{g})$ that deviates significantly from the hashing objective $L(\boldsymbol{h})$, because they cannot keep the codes exactly binary after convergence. Denote by $Q(\boldsymbol{g}, \boldsymbol{h})$ the quantization error function by binarizing continuous codes $\boldsymbol{g}$ into binary codes $\boldsymbol{h}$. Prior methods control the quantization error in two ways: **(a)** $\min L(\boldsymbol{g}) + Q(\boldsymbol{g}, \boldsymbol{h})$ through continuous optimization [44, 22]; **(b)** $\min L(\boldsymbol{h}) + Q(\boldsymbol{g}, \boldsymbol{h})$ through discrete optimization on $L(\boldsymbol{h})$ but continuous optimization on $Q(\boldsymbol{g}, \boldsymbol{h})$ (the continuous optimization is used for out-of-sample extension as discrete optimization cannot be extended to the test data) [24]. However, since $Q(\boldsymbol{g}, \boldsymbol{h})$ cannot be minimized to zero, there is a large gap between continuous codes and binary codes. To directly optimize $\min L(\boldsymbol{h})$, we must adopt *sign* as the *activation* function *within* deep networks, which enables generation of exactly binary codes but introduces the *ill-posed gradient* problem. This work is the first effort to learn sign-activated deep networks by continuation method, which can directly optimize $L(\boldsymbol{h})$ for deep learning to hash.

3. HashNet

In similarity retrieval systems, we are given a training set of N points $\{\boldsymbol{x}_i\}_{i=1}^N$, each represented by a D-dimensional feature vector $\boldsymbol{x}_i \in \mathbb{R}^D$. Some pairs of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are provided with similarity labels s_{ij}, where $s_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar while $s_{ij} = 0$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. The goal of deep learning to hash is to learn nonlinear hash function $f : \boldsymbol{x} \mapsto \boldsymbol{h} \in \{-1, 1\}^K$ from input space $\mathbb{R}^D$ to Hamming space $\{-1, 1\}^K$ using deep neural networks, which encodes each point $\boldsymbol{x}$ into compact K-bit binary hash code $\boldsymbol{h} = f(\boldsymbol{x})$ such that the similarity information between the given pairs $\mathcal{S}$ can be preserved in the compact hash codes. In supervised hashing, the similarity set $\mathcal{S} = \{s_{ij}\}$ can be constructed from semantic labels of data points or relevance feedback from click-through data in real retrieval systems.

To address the data imbalance and ill-posed gradient problems in an end-to-end learning framework, this paper presents **HashNet**, a novel architecture for deep learning to hash by continuation, shown in Figure 1. The architecture accepts pairwise input images $\{(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})\}$ and processes them through an end-to-end pipeline of deep representation learning and binary hash coding: (1) a convolutional network (CNN) for learning deep representation of each image $\boldsymbol{x}_i$, (2) a fully-connected hash layer (fch) for transforming the deep representation into K-dimensional representation $\boldsymbol{z}_i \in \mathbb{R}^K$, (3) a sign activation function $h = \operatorname{sgn}(z)$ for binarizing the K-dimensional representation $\boldsymbol{z}_i$ into K-bit

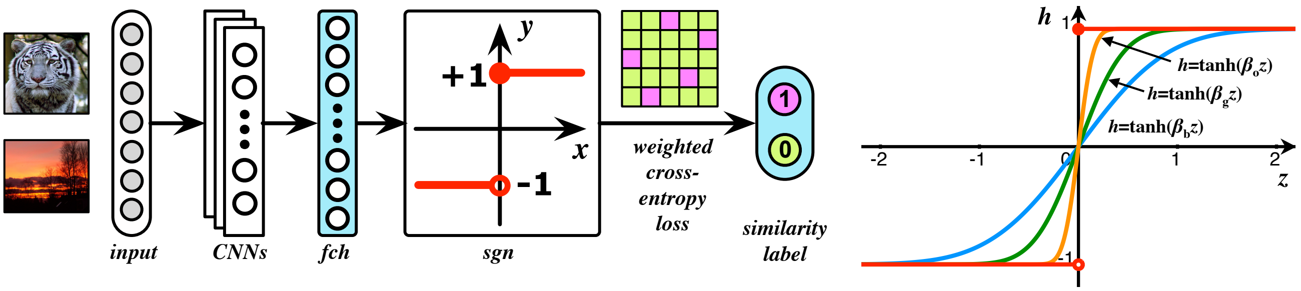


Figure 1. (left) The proposed HashNet for deep learning to hash by continuation, which is comprised of four key components: (1) Standard convolutional neural network (CNN), e.g. AlexNet and ResNet, for learning deep image representations, (2) a fully-connected hash layer (fch) for transforming the deep representation into K-dimensional representation, (3) a sign activation function (sgn) for binarizing the K-dimensional representation into K-bit binary hash code, and (4) a novel weighted cross-entropy loss for similarity-preserving learning from sparse data. (right) Plot of smoothed responses of the sign function $h = \operatorname{sgn}(z)$: Red is the sign function, and blue, green and orange show functions $h = \tanh(\beta z)$ with bandwidths $\beta_{\mathrm{b}} < \beta_{\mathrm{g}} < \beta_{\mathrm{o}}$. The key property is $\lim_{\beta \to \infty} \tanh(\beta z) = \operatorname{sgn}(z)$. *Best viewed in color.*

binary hash code $\boldsymbol{h}_i \in \{-1, 1\}^K$, and (4) a novel weighted cross-entropy loss for similarity-preserving learning from imbalanced data. We attack the ill-posed gradient problem of the non-smooth activation function $h = \operatorname{sgn}(z)$ by continuation, which starts with a smoothed activation function $y = \tanh(\beta x)$ and becomes more non-smooth by increasing β as the training proceeds, until eventually goes back to the original, difficult to optimize, sign activation function.

3.1. Model Formulation

To perform deep learning to hash from imbalanced data, we jointly preserve similarity information of pairwise images and generate binary hash codes by weighted maximum likelihood [6]. For a pair of binary hash codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, there exists a nice relationship between their Hamming distance $\operatorname{dist}_H(\cdot, \cdot)$ and inner product $\langle \cdot, \cdot \rangle$: $\operatorname{dist}_H(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{1}{2}(K - \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle)$. Hence, the Hamming distance and inner product can be used interchangeably for binary hash codes, and we adopt inner product to quantify pairwise similarity. Given the set of pairwise similarity labels $\mathcal{S} = \{s_{ij}\}$, the Weighted Maximum Likelihood (WML) estimation of the hash codes $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$ for all N training points is

$$\log P(\mathcal{S}|\boldsymbol{H}) = \sum_{s_{ij} \in \mathcal{S}} w_{ij} \log P(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j), \tag{1}$$

where $P(\mathcal{S}|\boldsymbol{H})$ is the weighted likelihood function, and w_{ij} is the weight for each training pair $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$, which is used to tackle the data imbalance problem by weighting the training pairs according to the importance of misclassifying that pair [6]. Since each similarity label in $\mathcal{S}$ can only be $s_{ij} = 1$ (similar) or $s_{ij} = 0$ (dissimilar), to account for the data imbalance between similar and dissimilar pairs, we set

$$w_{ij} = c_{ij} \cdot \begin{cases} |\mathcal{S}| \,/\, |\mathcal{S}_1|, & s_{ij} = 1 \\ |\mathcal{S}| \,/\, |\mathcal{S}_0|, & s_{ij} = 0 \end{cases} \tag{2}$$

where $\mathcal{S}_1 = \{s_{ij} \in \mathcal{S} : s_{ij} = 1\}$ is the set of similar pairs and $\mathcal{S}_0 = \{s_{ij} \in \mathcal{S} : s_{ij} = 0\}$ is the set of dissimilar pairs; c_{ij} is continuous similarity, i.e. $c_{ij} = \frac{\boldsymbol{y}_i \cap \boldsymbol{y}_j}{\boldsymbol{y}_i \cup \boldsymbol{y}_j}$ if labels $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are given, $c_{ij} = 1$ if only s_{ij} is given. For each pair, $P(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j)$ is the conditional probability of similarity label s_{ij} given a pair of hash codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, which can be naturally defined as pairwise logistic function,

$$\begin{aligned} P(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j) &= \begin{cases} \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle), & s_{ij} = 1 \\ 1 - \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle), & s_{ij} = 0 \end{cases} \\ &= \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle)^{s_{ij}} (1 - \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle))^{1 - s_{ij}} \end{aligned} \tag{3}$$

where $\sigma(x) = 1/(1 + e^{-\alpha x})$ is the *adaptive* sigmoid function with hyper-parameter α to control its bandwidth. Note that the sigmoid function with larger α will have larger saturation zone where its gradient is zero. To perform more effective back-propagation, we usually require $\alpha < 1$, which is more effective than the typical setting of $\alpha = 1$. Similar to logistic regression, we can see in pairwise logistic regression that the smaller the Hamming distance $\operatorname{dist}_H(\boldsymbol{h}_i, \boldsymbol{h}_j)$ is, the larger the inner product $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle$ as well as the conditional probability $P(1|\boldsymbol{h}_i, \boldsymbol{h}_j)$ will be, implying that pair $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ should be classified as similar; otherwise, the larger the conditional probability $P(0|\boldsymbol{h}_i, \boldsymbol{h}_j)$ will be, implying that pair $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ should be classified as dissimilar. Hence, Equation (3) is a reasonable extension of the logistic regression classifier to the pairwise classification scenario, which is optimal for binary similarity labels $s_{ij} \in \{0, 1\}$.

By taking Equation (3) into WML estimation in Equation (1), we achieve the optimization problem of HashNet,

$$\min_{\Theta} \sum_{s_{ij} \in \mathcal{S}} w_{ij} \left(\log\left(1 + \exp\left(\alpha \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle\right)\right) - \alpha s_{ij} \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle \right), \tag{4}$$

where Θ denotes the set of all parameters in deep networks. Note that, HashNet directly uses the sign activation function $\boldsymbol{h}_i = \operatorname{sgn}(\boldsymbol{z}_i)$ which converts the K-dimensional representation to *exactly* binary hash codes, as shown in Figure 1. By optimizing the WML estimation in Equation (4), we can enable deep learning to hash from imbalanced data under a statistically optimal framework. It is noteworthy that our work is the first attempt that extends the WML estimation from pointwise scenario to pairwise scenario. The HashNet can jointly preserve similarity information of pairwise images and generate *exactly* binary hash codes. Different from HashNet, previous deep-hashing methods need to first learn continuous embeddings, which are binarized in a separated step using the sign function. This will result in substantial quantization errors and significant losses of retrieval quality.

3.2. Learning by Continuation

HashNet learns *exactly* binary hash codes by converting the K-dimensional representation $\boldsymbol{z}$ of the hash layer fch, which is continuous in nature, to binary hash code $\boldsymbol{h}$ taking values of either $+1$ or -1. This binarization process can only be performed by taking the sign function $h = \operatorname{sgn}(z)$ as activation function on top of hash layer fch in HashNet,

$$h = \operatorname{sgn}(z) = \begin{cases} +1, & \text{if } z \geqslant 0 \\ -1, & \text{otherwise} \end{cases} \qquad (5)$$

Unfortunately, as the sign function is non-smooth and non-convex, its gradient is zero for all nonzero inputs, and is ill-defined at zero, which makes the standard back-propagation infeasible for training deep networks. This is known as the *vanishing gradient* problem, which has been a key difficulty in training deep neural networks via back-propagation [14].

Many optimization methods have been proposed to circumvent the vanishing gradient problem and enable effective network training with back-propagation, including unsupervised pre-training [14, 3], dropout [36], batch normalization [15], and deep residual learning [13]. In particular, Rectifier Linear Unit (ReLU) [29] activation function makes deep networks much easier to train and enables end-to-end learning algorithms. However, the sign activation function is so ill-defined that all the above optimization methods will fail. A very recent work, BinaryNet [5], focuses on training deep networks with activations constrained to $+1$ or -1. However, the training algorithm may be hard to converge as the feed-forward pass uses the sign activation (sgn) but the back-propagation pass uses a hard tanh (Htanh) activation. Optimizing deep networks with sign activation remains an open problem and a key challenge for deep learning to hash.

```
Algorithm 1: Optimizing HashNet by Continuation
  Input: A sequence $1 = \beta_0 < \beta_1 < \ldots < \beta_m = \infty$
  for stage $t = 0$ to $m$ do
      Train HashNet (4) with $\tanh(\beta_t z)$ as activation
      Set converged HashNet as next stage initialization
  end
  Output: HashNet with $\operatorname{sgn}(z)$ as activation, $\beta_m \to \infty$
```

This paper attacks the problem of non-convex optimization of deep networks with non-smooth sign activation by starting with a smoothed objective function which becomes more non-smooth as the training proceeds. It is inspired by recent studies in continuation methods [1], which address a complex optimization problem by smoothing the original function, turning it into a different problem that is easier to optimize. By gradually reducing the amount of smoothing during the training, it results in a sequence of optimization problems converging to the original optimization problem. Motivated by the continuation methods, we notice there exists a key relationship between the sign function and the scaled tanh function in the concept of limit in mathematics,

$$\lim_{\beta\to\infty} \tanh(\beta z) = \operatorname{sgn}(z), \qquad (6)$$

where $\beta > 0$ is a scaling parameter. Increasing β, the scaled tanh function $\tanh(\beta z)$ will become more non-smooth and more saturated so that the deep networks using $\tanh(\beta z)$ as the activation function will be more difficult to optimize, as in Figure 1 (right). But fortunately, as $\beta \to \infty$, the optimization problem will converge to the original deep learning to hash problem in (4) with $\operatorname{sgn}(z)$ activation function.

Using the continuation methods, we design an optimization method for HashNet in Algorithm 1. As deep network with $\tanh(z)$ as the activation function can be successfully trained, we start training HashNet with $\tanh(\beta_t z)$ as the activation function, where $\beta_0 = 1$. For each stage t, after HashNet converges, we increase β_t and train (i.e. fine-tune) HashNet by setting the converged network parameters as the initialization for training the HashNet in the next stage. By evolving $\tanh(\beta_t z)$ with $\beta_t \to \infty$, the network will converge to HashNet with $\operatorname{sgn}(z)$ as activation function, which can generate *exactly* binary hash codes as we desire. The efficacy of continuation in Algorithm 1 can be understood as multi-stage *pre-training*, i.e., pre-training HashNet with $\tanh(\beta_t z)$ activation function is used to initialize HashNet with $\tanh(\beta_{t+1} z)$ activation function, which enables easier progressive training of HashNet as the network is becoming non-smooth in later stages by $\beta_t \to \infty$. Using $m = 10$ we can already achieve fast convergence for training HashNet.

3.3. Convergence Analysis

We analyze that the continuation method in Algorithm 1 decreases HashNet loss (4) in each stage and each iteration. Let $L_{ij} = w_{ij}\left(\log\left(1 + \exp\left(\alpha\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle\right)\right) - \alpha s_{ij}\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle\right)$ and $L = \sum_{s_{ij}\in\mathcal{S}} L_{ij}$, where $\boldsymbol{h}_i \in \{-1, +1\}^K$ are *binary* hash codes. Note that when optimizing HashNet by continuation in Algorithm 1, the network activation in each stage t is $g = \tanh(\beta_t z)$, which is *continuous* in nature and will only become *binary* after convergence $\beta_t \to \infty$. Denote by $J_{ij} = w_{ij}\left(\log\left(1 + \exp\left(\alpha\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)\right) - \alpha s_{ij}\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)$ and $J = \sum_{s_{ij}\in\mathcal{S}} J_{ij}$ the true loss we optimize in Algorithm 1, where $\boldsymbol{g}_i \in \mathbb{R}^K$ and $\boldsymbol{h}_i = \operatorname{sgn}(\boldsymbol{g}_i)$. Our results are two theorems, with proofs provided in the supplemental materials.

Theorem 1. *The HashNet loss L will not change across stages t and $t+1$ with bandwidths switched from β_t to β_{t+1}.*

Theorem 2. *Loss L decreases when optimizing loss $J(\boldsymbol{g})$ by the stochastic gradient descent (SGD) within each stage.*

4. Experiments

We conduct extensive experiments to evaluate HashNet against several state-of-the-art hashing methods on three standard benchmarks. Datasets and implementations are available at http://github.com/thuml/HashNet.

4.1. Setup

The evaluation is conducted on three benchmark image retrieval datasets: ImageNet, NUS-WIDE and MS COCO.

ImageNet is a benchmark image dataset for Large Scale Visual Recognition Challenge (ILSVRC 2015) [32]. It contains over 1.2M images in the training set and 50K images in the validation set, where each image is single-labeled by one of the 1,000 categories. We randomly select 100 categories, use all the images of these categories in the training set as the database, and use all the images in the validation set as the queries; furthermore, we randomly select 100 images per category from the database as the training points.

NUS-WIDE[1] [4] is a public Web image dataset which contains 269,648 images downloaded from Flickr.com. Each image is manually annotated by some of the 81 ground truth concepts (categories) for evaluating retrieval models. We follow similar experimental protocols as DHN [44] and randomly sample 5,000 images as queries, with the remaining images used as the database; furthermore, we randomly sample 10,000 images from the database as training points.

MS COCO[2] [23] is an image recognition, segmentation, and captioning dataset. The current release contains 82,783 training images and 40,504 validation images, where each image is labeled by some of the 80 categories. After pruning images with no category information, we obtain 12,2218 images by combining the training and validation images. We randomly sample 5,000 images as queries, with the rest images used as the database; furthermore, we randomly sample 10,000 images from the database as training points.

Following standard evaluation protocol as previous work [40, 20, 44], the similarity information for hash function learning and for ground-truth evaluation is constructed from image labels: if two images i and j share at least one label, they are similar and $s_{ij} = 1$; otherwise, they are dissimilar and $s_{ij} = 0$. Note that, although we use the image labels to construct the similarity information, our proposed HashNet can learn hash codes when only the similarity information is available. By constructing the training data in this way, the ratio between the number of dissimilar pairs and the number of similar pairs is roughly 100, 5, and 1 for ImageNet, NUS-WIDE, and MS COCO, respectively. These datasets exhibit the data imbalance phenomenon and can be used to evaluate different hashing methods under data imbalance scenario.

We compare retrieval performance of **HashNet** with ten classical or state-of-the-art hashing methods: unsupervised methods **LSH** [10], **SH** [39], **ITQ** [12], supervised shallow methods **BRE** [19], **KSH** [25], **ITQ-CCA** [12], **SDH** [34], and supervised deep methods **CNNH** [40], **DNNH** [20], **DHN** [44]. We evaluate retrieval quality based on five standard evaluation metrics: Mean Average Precision (MAP), Precision-Recall curves (PR), Precision curves within Hamming distance 2 (P@H=2), Precision curves with respect to different numbers of top returned samples (P@N), and Histogram of learned codes without binarization. For fair comparison, all methods use identical training and test sets. We adopt MAP@1000 for ImageNet as each category has 1,300 images, and adopt MAP@5000 for the other datasets [44].

For shallow hashing methods, we use DeCAF$_7$ features [7] as input. For deep hashing methods, we use raw images as input. We adopt the AlexNet architecture [18] for all deep hashing methods, and implement HashNet based on the **Caffe** framework [17]. We fine-tune convolutional layers $conv1$–$conv5$ and fully-connected layers $fc6$–$fc7$ copied from the AlexNet model pre-trained on ImageNet 2012 and train the hash layer fch, all through back-propagation. As the fch layer is trained from scratch, we set its learning rate to be 10 times that of the lower layers. We use mini-batch stochastic gradient descent (SGD) with 0.9 momentum and the learning rate annealing strategy implemented in Caffe, and cross-validate the learning rate from 10^{-5} to 10^{-3} with a multiplicative step-size $10^{\frac{1}{2}}$. We fix the mini-batch size of images as 256 and the weight decay parameter as 0.0005.

4.2. Results

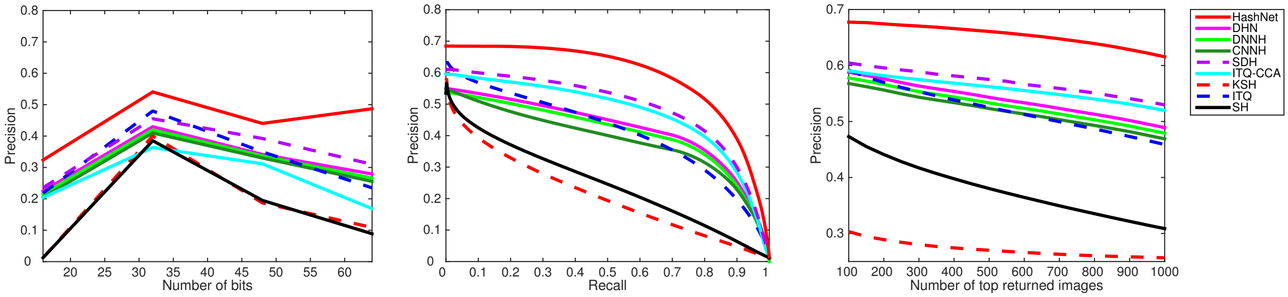
The Mean Average Precision (MAP) results are shown in Table 1. HashNet substantially outperforms all comparison methods. Specifically, compared to the best shallow hashing method using deep features as input, ITQ/ITQ-CCA, we achieve absolute boosts of 15.7%, 15.5%, and 9.1% in average MAP for different bits on ImageNet, NUS-WIDE, and MS COCO, respectively. Compared to the state-of-the-art deep hashing method, DHN, we achieve absolute boosts of 14.6%, 3.7%, 2.9% in average MAP for different bits on the three datasets, respectively. An interesting phenomenon is that the performance boost of HashNet over DHN is significantly different across the three datasets. Specifically, the performance boost on ImageNet is much larger than that on NUS-WIDE and MS COCO by about 10%, which is very impressive. Recall that the ratio between the number of dis-

[1] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

[2] http://mscoco.org

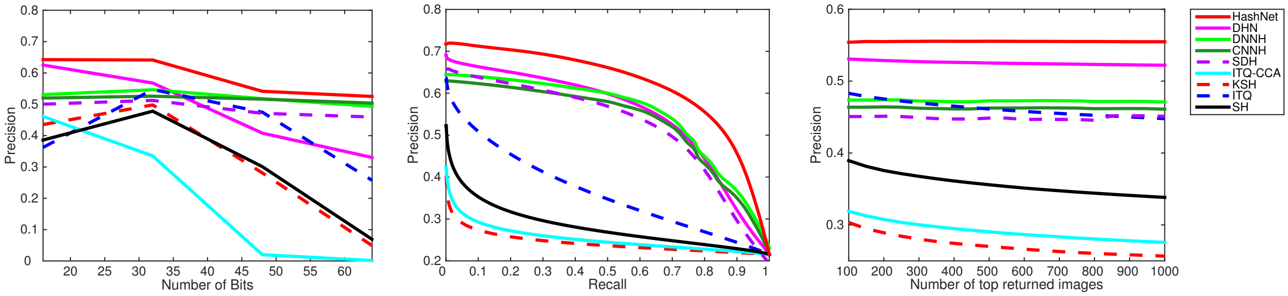
Table 1. Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on the Three Image Datasets

Method	ImageNet				NUS-WIDE				MS COCO			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
HashNet	**0.5059**	**0.6306**	**0.6633**	**0.6835**	**0.6623**	**0.6988**	**0.7114**	**0.7163**	**0.6873**	**0.7184**	**0.7301**	**0.7362**
DHN [44]	0.3106	0.4717	0.5419	0.5732	0.6374	0.6637	0.6692	0.6714	0.6774	0.7013	0.6948	0.6944
DNNH [20]	0.2903	0.4605	0.5301	0.5645	0.5976	0.6158	0.6345	0.6388	0.5932	0.6034	0.6045	0.6099
CNNH [40]	0.2812	0.4498	0.5245	0.5538	0.5696	0.5827	0.5926	0.5996	0.5642	0.5744	0.5711	0.5671
SDH [34]	0.2985	0.4551	0.5549	0.5852	0.4756	0.5545	0.5786	0.5812	0.5545	0.5642	0.5723	0.5799
KSH [25]	0.1599	0.2976	0.3422	0.3943	0.3561	0.3327	0.3124	0.3368	0.5212	0.5343	0.5343	0.5361
ITQ-CCA [12]	0.2659	0.4362	0.5479	0.5764	0.4598	0.4052	0.3732	0.3467	0.5659	0.5624	0.5297	0.5019
ITQ [12]	0.3255	0.4620	0.5170	0.5520	0.5086	0.5425	0.5580	0.5611	0.5818	0.6243	0.6460	0.6574
BRE [19]	0.0628	0.2525	0.3300	0.3578	0.5027	0.5290	0.5475	0.5546	0.5920	0.6224	0.6300	0.6336
SH [39]	0.2066	0.3280	0.3951	0.4191	0.4058	0.4209	0.4211	0.4104	0.4951	0.5071	0.5099	0.5101
LSH [10]	0.1007	0.2350	0.3121	0.3596	0.3283	0.4227	0.4333	0.5009	0.4592	0.4856	0.5440	0.5849



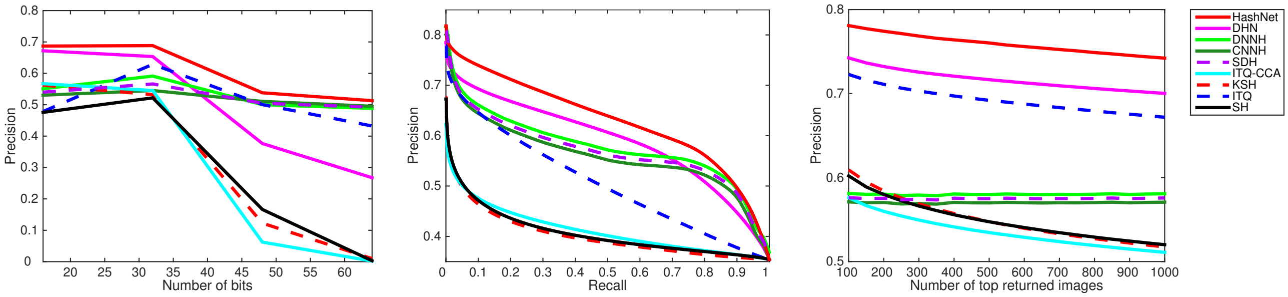
(a) Precision within Hamming radius 2 (b) Precision-recall curve @ 64 bits (c) Precision curve w.r.t. top-N @ 64 bits

Figure 2. The experimental results of HashNet and comparison methods on the ImageNet dataset under three evaluation metrics.



(a) Precision within Hamming radius 2 (b) Precision-recall curve @ 64 bits (c) Precision curve w.r.t. top-N @ 64 bits

Figure 3. The experimental results of HashNet and comparison methods on the NUS-WIDE dataset under three evaluation metrics.



(a) Precision within Hamming radius 2 (b) Precision-recall curve @ 64 bits (c) Precision curve w.r.t. top-N @ 64 bits

Figure 4. The experimental results of HashNet and comparison methods on the MS COCO dataset under three evaluation metrics.

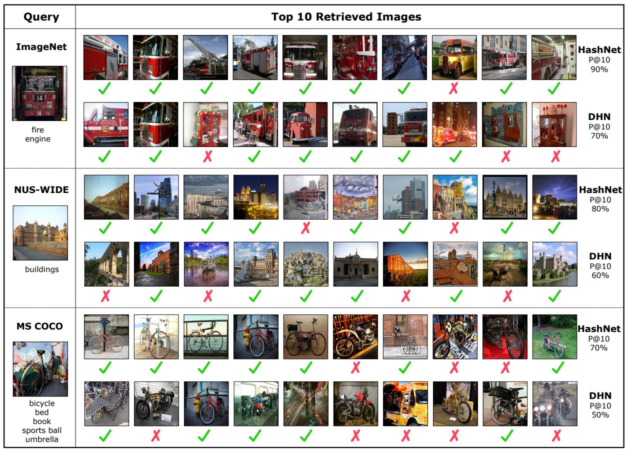


Figure 5. Examples of top 10 retrieved images and precision@10.

similar pairs and the number of similar pairs is roughly 100, 5, and 1 for ImageNet, NUS-WIDE and MS COCO, respectively. This data imbalance problem substantially deteriorates the performance of hashing methods trained from pairwise data, including all the deep hashing methods. HashNet enhances deep learning to hash from imbalanced dataset by Weighted Maximum Likelihood (WML), which is a principled solution to tackling the data imbalance problem. This lends it the superior performance on imbalanced datasets.

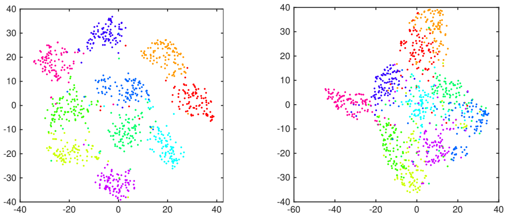
The performance in terms of Precision within Hamming radius 2 (P@H=2) is very important for efficient retrieval with binary hash codes since such Hamming ranking only requires $O(1)$ time for each query. As shown in Figures 2(a), 3(a) and 4(a), HashNet achieves the highest P@H=2 results on all three datasets. In particular, P@H=2 of HashNet with 32 bits is better than that of DHN with any bits. This validates that HashNet can learn more compact binary codes than DHN. When using longer codes, the Hamming space will become sparse and few data points fall within the Hamming ball with radius 2 [9]. This is why most hashing methods achieve best accuracy with moderate code lengths.

The retrieval performance on the three datasets in terms of Precision-Recall curves (PR) and Precision curves with respect to different numbers of top returned samples (P@N) are shown in Figures 2(b)∼4(b) and Figures 2(c)∼4(c), respectively. HashNet outperforms comparison methods by large margins. In particular, HashNet achieves much higher precision at lower recall levels or when the number of top results is small. This is desirable for precision-first retrieval, which is widely implemented in practical systems. As an intuitive illustration, Figure 5 shows that HashNet can yield much more relevant and user-desired retrieval results.

Recent work [28] studies two evaluation protocols for supervised hashing: (1) supervised retrieval protocol where queries and database have identical classes and (2) zero-shot retrieval protocol where queries and database have different classes. Some supervised hashing methods perform well in one protocol but poorly in another protocol. Table 2 shows the MAP results on ImageNet dataset under the zero-shot retrieval protocol, where HashNet substantially outperforms DHN. Thus, HashNet works well under different protocols.

Table 2. MAP on ImageNet with Zero-Shot Retrieval Protocol [28]

Method	16 bits	32 bits	48 bits	64 bits
HashNet	**0.4411**	**0.5274**	**0.5651**	**0.5756**
DHN [44]	0.2891	0.4421	0.5123	0.5342



(a) HashNet (b) DHN

Figure 6. The t-SNE of hash codes learned by HashNet and DHN.

4.3. Empirical Analysis

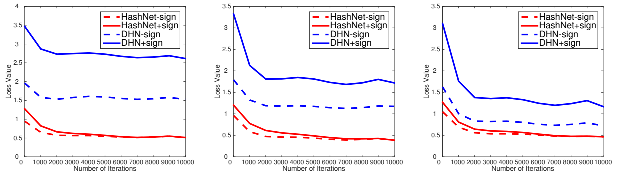
Visualization of Hash Codes: We visualize the t-SNE [7] of hash codes generated by HashNet and DHN on ImageNet in Figure 6 (for ease of visualization, we sample 10 categories). We observe that the hash codes generated by HashNet show clear discriminative structures in that different categories are well separated, while the hash codes generated by DHN do not show such discriminative structures. This suggests that HashNet can learn more discriminative hash codes than DHN for more effective similarity retrieval.

Ablation Study: We go deeper with the efficacy of the weighted maximum likelihood and continuation methods. We investigate three variants of HashNet: (1) **HashNet+C**, variant using continuous similarity $c_{ij} = \frac{\boldsymbol{y}_i \cap \boldsymbol{y}_j}{\boldsymbol{y}_i \cup \boldsymbol{y}_j}$ when image labels are given; (2) **HashNet-W**, variant using maximum likelihood instead of weighted maximum likelihood, i.e. $w_{ij} = 1$; (3) **HashNet-sgn**, variant using $\tanh()$ instead of $\mathrm{sgn}()$ as activation function to generate continuous codes and requiring a separated binarization step to generate hash codes. We compare results of these variants in Table 3.

By weighted maximum likelihood estimation, HashNet outperforms HashNet-W by substantially large margins of 12.4%, 2.8% and 0.1% in average MAP for different bits on ImageNet, NUS-WIDE and MS COCO, respectively. The standard maximum likelihood estimation has been widely adopted in previous work [40, 44]. However, this estimation does not account for the data imbalance, and may suffer from performance drop when training data is highly imbalanced (e.g. ImageNet). In contrast, the proposed weighted maximum likelihood estimation (1) is a principled solution to tackling the data imbalance problem by weighting the training pairs according to the importance of misclassifying that pair. Recall that MS COCO is a balanced dataset, hence HashNet and HashNet-W may yield similar MAP results.

Table 3. Mean Average Precision (MAP) Results of HashNet and Its Variants, HashNet+C, HashNet-W, and HashNet-sgn on Three Datasets

Method	ImageNet				NUS-WIDE				MS COCO			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
HashNet+C	**0.5059**	**0.6306**	**0.6633**	**0.6835**	**0.6646**	**0.7024**	**0.7209**	**0.7259**	**0.6876**	**0.7261**	**0.7371**	**0.7419**
HashNet	**0.5059**	**0.6306**	**0.6633**	**0.6835**	0.6623	0.6988	0.7114	0.7163	0.6873	0.7184	0.7301	0.7362
HashNet-W	0.3350	0.4852	0.5668	0.5992	0.6400	0.6638	0.6788	0.6933	0.6853	0.7174	0.7297	0.7348
HashNet-sgn	0.4249	0.5450	0.5828	0.6061	0.6603	0.6770	0.6921	0.7020	0.6449	0.6891	0.7056	0.7138



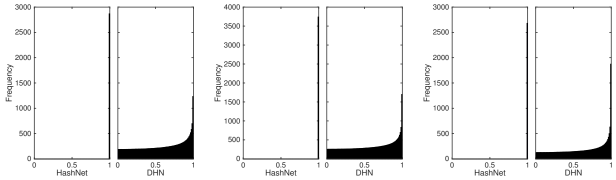
(a) ImageNet (b) NUS-WIDE (c) COCO

Figure 7. Losses of HashNet and DHN through training process.

By further considering continuous similarity ($c_{ij} = \frac{\boldsymbol{y}_i \cap \boldsymbol{y}_j}{\boldsymbol{y}_i \cup \boldsymbol{y}_j}$), HashNet+C achieves even better accuracy than HashNet.

By training HashNet with continuation, HashNet outperforms HashNet-sgn by substantial margins of 8.1%, 1.4% and 3.0% in average MAP on ImageNet, NUS-WIDE, and MS COCO, respectively. Due to the ill-posed gradient problem, existing deep hashing methods cannot learn exactly binary hash codes using $\operatorname{sgn}()$ as activation function. Instead, they need to use surrogate functions of $\operatorname{sgn}()$, e.g. $\tanh()$, as the activation function and learn continuous codes, which require a separated binarization step to generate hash codes. The proposed continuation method is a principled solution to deep learning to hash with $\operatorname{sgn}()$ as activation function, which learn lossless binary hash codes for accurate retrieval.

Loss Value Through Training Process: We compare the change of loss values of HashNet and DHN through the training process on ImageNet, NUS-WIDE and MSCOCO. We display the loss values before (-sign) and after (+sign) binarization, i.e. $J(\boldsymbol{g})$ and $L(\boldsymbol{h})$. Figure 7 reveals three important observations: **(a)** Both methods converge in terms of the loss values before and after binarization, which validates the convergence analysis in Section 3.3. **(b)** HashNet converges with a much smaller training loss than DHN both before and after binarization, which implies that HashNet can preserve the similarity relationship in *Hamming* space much better than DHN. **(c)** The two loss curves of HashNet before and after binarization become close to each other and overlap completely when convergence. This shows that the continuation method enables HashNet to approach the true loss defined on the exactly binary codes without continuous relaxation. But there is a large gap between two loss curves of DHN, implying that DHN and similar methods [34, 22, 24] cannot learn exactly binary codes by minimizing quantization error of codes before and after binarization.



(a) ImageNet (b) NUS-WIDE (c) COCO

Figure 8. Histogram of non-binarized codes of HashNet and DHN.

Histogram of Codes Without Binarization: As discussed previously, the proposed HashNet can learn exactly binary hash codes while previous deep hashing methods can only learn continuous codes and generate binary hash codes by post-step sign thresholding. To verify this key property, we plot the histograms of codes learned by HashNet and DHN on the three datasets without post-step binarization. The histograms can be plotted by evenly dividing $[0, 1]$ into 100 bins, and calculating the frequency of codes falling into each bin. To make the histograms more readable, we show absolute code values (x-axis) and squared root of frequency (y-axis). Histograms in Figure 8 show that DHN can only generate continuous codes spanning across the whole range of $[0, 1]$. This implies that if we quantize these continuous codes into binary hash codes (taking values in $\{-1, 1\}$) in a post-step, we may suffer from large quantization error especially for the codes near zero. On the contrary, the codes of HashNet without binarization are already exactly binary.

5. Conclusion

This paper addressed deep learning to hash from imbalanced similarity data by the continuation method. The proposed HashNet can learn exactly binary hash codes by optimizing a novel weighted pairwise cross-entropy loss function in deep convolutional neural networks. HashNet can be effectively trained by the proposed multi-stage pre-training algorithm carefully crafted from the continuation method. Comprehensive empirical evidence shows that HashNet can generate exactly binary hash codes and yield state-of-the-art multimedia retrieval performance on standard benchmarks.

6. Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2016YFB1000701), the National Natural Science Foundation of China (No. 61502265, 61325008, and 71690231), the National Sci.&Tech. Supporting Program (2015BAF32B01), and the Tsinghua TNList Projects.

References

[1] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012. 2, 4

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, Aug 2013. 1

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 153–160. MIT Press, 2007. 4

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *ICMR*. ACM, 2009. 5

[5] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. In *NIPS*, 2016. 4

[6] J. P. Dmochowski, P. Sajda, and L. C. Parra. Maximum likelihood in cost-sensitive learning: Model specification, approximations, and upper bounds. *Journal of Machine Learning Research (JMLR)*, 11(Dec):3313–3332, 2010. 3

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 5, 7

[8] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483. IEEE, 2015. 1

[9] D. J. Fleet, A. Punjani, and M. Norouzi. Fast search in hamming space with multi-index hashing. In *CVPR*. IEEE, 2012. 1, 7

[10] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529. ACM, 1999. 1, 5, 6

[11] Y. Gong, S. Kumar, H. Rowley, S. Lazebnik, et al. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, pages 484–491. IEEE, 2013. 1

[12] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011. 1, 2, 5, 6

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 2, 4

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 1, 4

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[16] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(1):117–128, Jan 2011. 2

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia Conference*. ACM, 2014. 5

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 5

[19] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009. 1, 2, 5, 6

[20] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*. IEEE, 2015. 1, 2, 5, 6

[21] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, Feb. 2006. 1

[22] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 2016. 1, 2, 8

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 5

[24] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016. 1, 2, 8

[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*. IEEE, 2012. 1, 2, 5, 6

[26] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*. ACM, 2011. 2

[27] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *CVPR*, pages 1570–1577. IEEE, 2013. 1

[28] C. Ma, I. W. Tsang, F. Peng, and C. Liu. Partial hash update via hamming subspace learning. *IEEE Transactions on Image Processing (TIP)*, 26(4):1939–1951, 2017. 7

[29] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *ICML*, pages 807–814. Omnipress, 2010. 4

[30] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360. ACM, 2011. 1, 2

[31] M. Norouzi, D. M. Blei, and R. R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, pages 1061–1069, 2012. 2

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 5

[33] R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, pages 412–419, 2007. 2

[34] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*. IEEE, June 2015. 1, 5, 6, 8

[35] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(12):1349–1380, 2000. 2

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, Jan. 2014. 4

[37] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(12):2393–2406, 2012. 1

[38] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. Arxiv, 2014. 1, 2

[39] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009. 2, 5, 6

[40] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, pages 2156–2162. AAAI, 2014. 1, 2, 5, 6, 7

[41] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, pages 353–360. ACM, 2014. 1

[42] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In *SIGIR*, pages 173–182. ACM, 2014. 1

[43] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015. 1

[44] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*. AAAI, 2016. 1, 2, 5, 6, 7

A. Supplemental Material: HashNet: Deep Learning to Hash by Continuation

A.1. Convergence Analysis

We briefly analyze that the continuation optimization in Algorithm 1 will decrease the loss of HashNet (4) in each stage and in each iteration until converging to HashNet with sign activation function that generates *exactly* binary codes.

Let $L_{ij} = w_{ij}\left(\log\left(1+\exp\left(\alpha\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle\right)\right) - \alpha s_{ij}\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle\right)$ and $L = \sum_{s_{ij}\in\mathcal{S}} L_{ij}$, where $\boldsymbol{h}_i \in \{-1,+1\}^K$ are *binary* hash codes. Note that when optimizing HashNet by continuation in Algorithm 1, network activation in each stage t is $g = \tanh(\beta_t z)$, which is *continuous* in nature and will only become *binary* when convergence $\beta_t \to \infty$. Denote by $J_{ij} = w_{ij}\left(\log\left(1+\exp\left(\alpha\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)\right) - \alpha s_{ij}\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)$ and $J = \sum_{s_{ij}\in\mathcal{S}} J_{ij}$ the true loss we optimize in Algorithm 1, where $\boldsymbol{g}_i \in \mathbb{R}^K$ and note that $\boldsymbol{h}_i = \operatorname{sgn}(\boldsymbol{g}_i)$. We will show that HashNet loss $L(\boldsymbol{h})$ descends when minimizing $J(\boldsymbol{g})$.

Theorem 3. *The HashNet loss L will not change across stages t and $t+1$ with bandwidths switched from β_t to β_{t+1}.*

Proof. When the algorithm switches from stages t to $t+1$ with bandwidths changed from β_t to β_{t+1}, only the network activation is changed from $\tanh(\beta_t z)$ to $\tanh(\beta_{t+1} z)$ but its sign $h = \operatorname{sgn}(\tanh(\beta_t z)) = \operatorname{sgn}(\tanh(\beta_{t+1} z))$, i.e. the hash code, remains the same. Thus L is unchanged. □

For each pair of binary codes $\boldsymbol{h}_i, \boldsymbol{h}_j$ and their continuous counterparts $\boldsymbol{g}_i, \boldsymbol{g}_j$, the derivative of J w.r.t. each bit k is

$$\frac{\partial J}{\partial g_{ik}} = w_{ij}\alpha\left(\frac{1}{1+\exp\left(-\alpha\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)} - s_{ij}\right) g_{jk}, \tag{7}$$

where $k = 1, \ldots, K$. The derivative of J w.r.t. $\boldsymbol{g}_j$ can be defined similarly. Updating $\boldsymbol{g}_i$ by SGD, the updated $\boldsymbol{g}_i'$ is

$$\begin{aligned} g_{ik}' &= g_{ik} - \eta\frac{\partial J}{\partial g_{ik}} \\ &= g_{ik} - \eta w_{ij}\alpha\left(\frac{1}{1+\exp\left(-\alpha\left\langle \boldsymbol{g}_i, \boldsymbol{g}_j\right\rangle\right)} - s_{ij}\right) g_{jk}, \end{aligned} \tag{8}$$

where η is the learning rate and $\boldsymbol{g}_j'$ is computed similarly.

Lemma 1. *Denote by $\boldsymbol{h}_i = \operatorname{sgn}(\boldsymbol{g}_i)$, $\boldsymbol{h}_i' = \operatorname{sgn}(\boldsymbol{g}_i')$, then*

$$\begin{cases} \left\langle \boldsymbol{h}_i', \boldsymbol{h}_j'\right\rangle \geqslant \left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle, & s_{ij} = 1, \\ \left\langle \boldsymbol{h}_i', \boldsymbol{h}_j'\right\rangle \leqslant \left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle, & s_{ij} = 0. \end{cases} \tag{9}$$

Proof. Since $\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle = \sum_{k=1}^K h_{ik}h_{jk}$, Lemma 1 can be proved by verifying that $h_{ik}'h_{jk}' \geqslant h_{ik}h_{jk}$ if $s_{ij} = 1$ and $h_{ik}'h_{jk}' \leqslant h_{ik}h_{jk}$ if $s_{ij} = 0, \forall k = 1, 2, \ldots, K$.

Case 1. $s_{ij} = 0$.

(1) If $g_{ik} < 0$, $g_{jk} > 0$, then $\frac{\partial J}{\partial g_{ik}} > 0$, $\frac{\partial J}{\partial g_{jk}} < 0$. Thus, $h_{ik}' \leqslant h_{ik} = -1$, $h_{jk}' \geqslant h_{jk} = 1$. And we have $h_{ik}'h_{jk}' = -1 = h_{ik}h_{jk}$.

(2) If $g_{ik} > 0$, $g_{jk} < 0$, then $\frac{\partial J}{\partial g_{ik}} < 0$, $\frac{\partial J}{\partial g_{jk}} > 0$. Thus, $h_{ik}' \geqslant h_{ik} = 1$, $h_{jk}' \leqslant h_{jk} = -1$. And we have $h_{ik}'h_{jk}' = -1 = h_{ik}h_{jk}$.

(3) If $g_{ik} < 0$, $g_{jk} < 0$, then $\frac{\partial J}{\partial g_{ik}} < 0$, $\frac{\partial J}{\partial g_{jk}} < 0$. Thus $h_{ik}' \geqslant h_{ik} = -1$, $h_{jk}' \geqslant h_{jk} = -1$. So h_{ik}' and h_{jk}' may be either $+1$ or -1 and we have $h_{ik}'h_{jk}' \leqslant 1 = h_{ik}h_{jk}$.

(4) If $g_{ik} > 0$, $g_{jk} > 0$, then $\frac{\partial J}{\partial g_{ik}} > 0$, $\frac{\partial J}{\partial g_{jk}} > 0$. Thus $h_{ik}' \leqslant h_{ik} = 1$, $h_{jk}' \leqslant h_{jk} = 1$. So h_{ik}' and h_{jk}' may be either $+1$ or -1 and we have $h_{ik}'h_{jk}' \leqslant 1 = h_{ik}h_{jk}$.

Case 2. $s_{ij} = 1$. *It can be proved similarly as Case 1.* □

Theorem 4. *Loss L decreases when optimizing loss $J(\boldsymbol{g})$ by the stochastic gradient descent (SGD) within each stage.*

Proof. The gradient of loss L w.r.t. *hash* codes $\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle$ is

$$\frac{\partial L}{\partial\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle} = w_{ij}\alpha\left(\frac{1}{1+\exp\left(-\alpha\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle\right)} - s_{ij}\right). \tag{10}$$

We observe that

$$\begin{cases} \frac{\partial L}{\partial\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle} \leqslant 0, & s_{ij} = 1, \\ \frac{\partial L}{\partial\left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle} \geqslant 0, & s_{ij} = 0. \end{cases} \tag{11}$$

By substituting Lemma 1: if $s_{ij} = 1$, then $\left\langle \boldsymbol{h}_i', \boldsymbol{h}_j'\right\rangle \geqslant \left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle$, and thus $L(\boldsymbol{h}_i', \boldsymbol{h}_j') \leqslant L(\boldsymbol{h}_i, \boldsymbol{h}_j)$; if $s_{ij} = 0$, then $\left\langle \boldsymbol{h}_i', \boldsymbol{h}_j'\right\rangle \leqslant \left\langle \boldsymbol{h}_i, \boldsymbol{h}_j\right\rangle$, and thus $L(\boldsymbol{h}_i', \boldsymbol{h}_j') \leqslant L(\boldsymbol{h}_i, \boldsymbol{h}_j)$. □