Deep Polarized Network for Supervised Learning of Accurate Binary Hashing Codes

Lixin Fan[1] , **Kam Woh Ng**[1] , **Ce Ju**[1] , **Tianyu Zhang**[1] , **Chee Seng Chan**[2]
[1]WeBank AI, Shenzhen, China
[2]University of Malaya, Kuala Lumpur, Malaysia
{lixinfan, jinhewu, ceju, brutuszhang}@webank.com, cs.chan@um.edu.my

Abstract

This paper proposes a novel *deep polarized network* (DPN) for learning to hash, in which each channel in the network outputs is pushed far away from zero by employing a differentiable bit-wise hinge-like loss which is dubbed as *polarization loss*. Reformulated within a generic Hamming Distance Metric Learning framework [Norouzi *et al.*, 2012], the proposed polarization loss bypasses the requirement to prepare *pairwise* labels for (dis-)similar items and, yet, the proposed loss strictly bounds from above the pairwise Hamming Distance based losses. The intrinsic connection between *pairwise* and *pointwise* label information, as disclosed in this paper, brings about the following methodological improvements: (a) we may directly employ the proposed *differentiable polarization loss* with no large deviations incurred from the target Hamming distance based loss; and (b) the subtask of assigning binary codes becomes extremely simple — even *random codes* assigned to each class suffice to result in state-of-the-art performances, as demonstrated in CIFAR10, NUS-WIDE and ImageNet100 datasets.

1 Introduction

The ultimate goal of *learning to hash* (LtH) is to *preserve the similarity* of neighboring data points such that semantically similar items are grouped together and can be efficiently retrieved upon querying requests. Generally, the learning objective is often formulated as the minimization of a loss based on the *pairwise* hash code differences e.g. in [Lin *et al.*, 2013]:

$$\min_{\Phi(\cdot)} \sum_{i,j} L\big(\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j), y_{ij}\big), \tag{1}$$

in which $y_{ij} = \{-1, +1\}$ is the given similar/dissimilar relations for a sample data pairs $(\mathbf{x}_i, \mathbf{x}_j)$, the loss $L(\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j), y_{ij})$ simultaneously minimizes intra-class and maximizes inter-class Hamming distances between similar/dissimilar hash codes, and $\Phi : \mathcal{X} \rightarrow \mathbb{H}^b$ are the *binary* hash functions to be learned.

One central difficulty for optimizing (Eq. 1) lies in the *vanishing gradients* of binary hashing functions $\Phi(\mathbf{x}_i)$ which in general lead to a NP-hard binary optimization problem. Two principled solutions have been proposed in the literature — the first solution is to approximate Φ with continuous relaxation e.g. either by evolving a smoothed activation function [Cao *et al.*, 2017] or simply eliminating the hard binary constrains and penalizing large deviations from the desired binary outcomes [Liu *et al.*, 2016]. This continuous relaxation allows gradient-based optimization methods e.g. back-propagation to be directly applied, yet deviations from binary codes inevitably induces *quantization errors* and arguably gives rise to sub-optimal hash codes [Su *et al.*, 2018]. The second solution is to decompose the original optimization problem into two sub-problems i.e. *binary code inference* and *hash function learning* [Lin *et al.*, 2013]. Although the optimal hash functions can be learned with respect to a variety of loss functions defined on the recovered binary codes, the assignment of appropriate codes to each data points in the initial step is itself a non-trivial task and sophisticated algorithms e.g. bit-wise *block coordinate descent* or *discrete cyclic coordinate descent* have to be employed [Lin *et al.*, 2013; Shen *et al.*, 2015]. Bearing in mind the merits and shortcomings of these two aforementioned solutions, one may wonder whether it is possible to come up with a simple *differentiable loss* that is guaranteed to minimize the original Hamming distance based loss with no quantization errors incurred, and at the same time, no sophisticated binary optimization solvers are needed. This open problem is the primary motivation of our work illustrated in this paper.

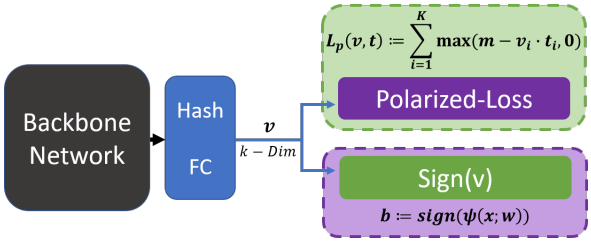

Figure 1: The proposed DPN architectures for *training* and *inference*. See Section 3.1 for the backbone network used.

To this end, the present paper proposes a novel *deep polarized network* (DPN) for learning to hash as illustrated in Figure 1, in which each channel in the network outputs is enforced towards either positive or negative extremes by employing a differentiable bit-wise hinge-like loss which we

Labels	Handcraft	Pre-trained
Pointwise	[Gong and Lazebnik, 2011] [Shen *et al.*, 2015]	[Yang *et al.*, 2018] [Shen *et al.*, 2019]
Pairwise	[Kulis and Darrell, 2009] [Liu *et al.*, 2012] [Lin *et al.*, 2013]	[Li *et al.*, 2016] [Cakir *et al.*, 2018] [Cakir *et al.*, 2019]
Triplet & Rank	[Norouzi *et al.*, 2012] [Wang *et al.*, 2013] [Wang *et al.*, 2015]	[Yao *et al.*, 2016] [Liu *et al.*, 2018] [He *et al.*, 2018]

Table 1: Categorization of LtH methods and representative losses. **Handcraft** denotes methods do not use neural networks for feature extraction; **Pre-trained** denotes methods use CNN image features pre-trained for classification task; For *pointwise* labels, often the classification objectives were used e.g. [Shen *et al.*, 2019]. For *pairwise* labels, losses push similar pairs together while dissimilar ones apart [Liu *et al.*, 2016]. For *triplet & rank* labels, often the triplet-based hinge loss variants were used [Norouzi *et al.*, 2012]. Please refer to [Li *et al.*, 2016; Lin *et al.*, 2017; Wang *et al.*, 2018; Cakir *et al.*, 2019] for thorough reviews of related works.

dubbed as *"polarization loss"*. A number of methodological advantages of the proposed polarized networks are as follows. First, it is demonstrated by both theoretical proof and empirical investigations, that the proposed polarization loss *strictly bounds* from above the Hamming distance between similar/dissimilar hash code pairs. As such, minimizing the polarization loss is equivalent to simultaneous minimizing the intra-class and maximizing inter-class Hamming distances (see Proposition 3). Second, the subtask of binary code inferences in the two-step approach becomes extremely simple — even *random codes* assigned to each class suffice to result in state-of-the-art performances, as demonstrated in CIFAR10, NUS-WIDE and ImageNet100 datasets. Moreover, hash codes accuracies can be consistently elevated above the-state-of-the-art, by encoding non-polarized network outputs with an intermediate state i.e. 0 between $\{-1, +1\}$. This ternary assignment results in 2% to 8% improvements of mAP for experiments in all three datasets.

1.1 Related Work

Numerous hashing methods have been proposed over the years and, due to limited space, we only review those related to our method. Please refer to [Li *et al.*, 2016; Lin *et al.*, 2017; Cakir *et al.*, 2018; Wang *et al.*, 2018; Cakir *et al.*, 2019] for thorough reviews of related works. As shown in Table 1, we broadly categorize existing hashing methods according to two dimensions i.e. (i) what type of label information is exploited, and (ii) in which way low-level (image) features are extracted.

Among the four different types of label information i.e. *pointwise, pairwise, triplet* and *rank* labels that were used for LtH, the pairwise labels is probably the most commonly employed e.g. in [Kulis and Darrell, 2009; Jeong and Song, 2018; Cakir *et al.*, 2019], with computational complexity $O(N^2)$ for N data points. In terms of neural network architectures, Siamese networks are often needed to take pairs or triplets images as inputs during the learning stage [Li *et al.*, 2016; Wang *et al.*, 2016; Zhuang *et al.*, 2016]. In contrast, pointwise labels lead to loss functions which are faster to compute and easier to optimize. Although pointwise labels enjoyed computational and network simplicities e.g. as shown in [Shen *et al.*, 2019], previous works did not disclose the intrinsic connection between pointwise and pairwise losses, which is illustrated in Section 2.2 of this paper.

Recent LtH methods have witnessed significant improvements in terms of hashing accuracies by replacing handcrafted (image) features with Deep CNN features which are first pre-trained for classification task e.g. with ImageNet dataset and subsequently fine-tuned for hashing [Lin *et al.*, 2015; Zhao *et al.*, 2015; Cao *et al.*, 2018]. It was also demonstrated e.g. by [Shen *et al.*, 2015; Yang *et al.*, 2018] that *jointly* learning features with hash functions may improve the hashing accuracies.

A hinge-like loss (Eq. 4) with a Hamming Ball radius parameter was adopted in [Norouzi and Blei, 2011] to differentiate similar/dissimilar points. In a similar vein, hinge-like losses were also adopted for triplet or rank based labels e.g. within the well-developed Hamming Distance Metric Learning framework [Norouzi *et al.*, 2012]. However all these hinge-like losses are defined in the binary Hamming space, which are different from the polarization loss defined on real-valued network outputs (see definition in Section 2.2).

The proposed polarization network bears some similarities with the binary weight networks used for network compression e.g. [Hu *et al.*, 2018]. Also a sign loss was used to embed signatures into deep neural networks to provide ownership verification for IP protection [Fan *et al.*, 2019]. Nevertheless, motivations of these applications were different from LtH and, moreover, the connection with pairwise hashing objectives disclosed in this paper was not mentioned there.

2 Deep Polarized Networks

This section illustrates a novel *deep polarized network* (DPN), in which the network outputs are pushed away from zero to either positive or negative sides. We first formulate below the DPN within the well developed Hamming Distance Metric Learning framework [Norouzi *et al.*, 2012] and then elaborate the proposed polarization loss, which, to our best knowledge, is the only *single differentiable loss term* that simultaneously minimizes inter-class and maximizes intra-class Hamming distances.

2.1 Deep Polarized Network for Hamming Distance Metric Learning

Our goal is to learn a hash mapping that maps real-valued data points $\mathbf{x} \in \mathbb{R}^p$ into K-bit binary codes $\mathbf{b} \in \mathbb{H}^K := \{-1, +1\}^K$, such that similar data are mapped to similar binary codes with small Hamming distances, and vice versa.

For K-bit binary codes $\mathbf{b}$, the hash mapping function $\mathbf{\Phi}$ is parameterized by $\mathbf{w}$,

$$\mathbf{b} = \mathbf{\Phi}(\mathbf{x}; \mathbf{w}) := \mathtt{sign}(\mathbf{\Psi}(\mathbf{x}; \mathbf{w})), \tag{2}$$

in which the general function $\mathbf{\Psi} : \mathbb{R}^p \to \mathbb{R}^K$ admits different forms of implementations as shown in the Hamming Distance Metric Learning framework [Norouzi *et al.*, 2012]. In this work, $\mathbf{\Psi}$ can be implemented in any existing deep CNN appended with a fully connected output layer that consists of

K channels representing K bits. See Figure 1 for a diagram of the DPN architecture.

The choice of loss function is crucial for learning good similarity measures. While majority of the existing LtH methods formulate learning objectives in terms of *pairwise* or *triplet* similarities e.g. in [Norouzi and Blei, 2011; Norouzi *et al.*, 2012], we show that *pairwise or triplet* label information are actually not needed for the polarization loss proposed below.

2.2 Polarization Loss

Definition 1 (Polarization loss)**.** For each data $\mathbf{x} \in \mathcal{X}$ and its corresponding output vector $\mathbf{v} := \Psi(\mathbf{x}; \mathbf{w}) \in \mathbb{R}^K$, the polarization loss is defined on the vector $\mathbf{v}$ with respect to a pre-set target binary code $\mathbf{t} \in \mathcal{H}$ as follows,

$$\mathcal{L}_{\mathrm{P}}(\mathbf{v}, \mathbf{t}) := \sum_{i=1}^{K} \max(m - v_i \cdot t_i, 0), \tag{3}$$

where the margin threshold is pre-set, $m \geq 1$, for the bound in Lemma 1 to be strict.

By minimizing the polarization loss (Eq. 3) during the learning phase, magnitudes of each DPN output channels are induced above the threshold m while corresponding signs are aligned to the target vector $\mathbf{t}$. Figure 5 illustrates the distribution of outputs $\mathbf{v}$, for example images fed to a DPN. Clearly large margins push the network outputs further away from zero. It must be noted that, the outputs for correctly coded images are polarized while non-polarized outputs are more likely observed for mis-classified ones.

Lemma 1. For output vector $\mathbf{v} = \Psi(\mathbf{x}; \mathbf{w})$, the Hamming distance $\mathcal{D}_h(\mathbf{b}, \mathbf{t}) := \frac{1}{2}(K - \mathbf{b} \cdot \mathbf{t})$ between K-bits binary hash code $\mathbf{b} = \mathtt{sign}(\mathbf{v})$ and the corresponding binary vector $\mathbf{t}$ is upper bounded by the *polarization loss*

$$\mathcal{D}_h(\mathbf{b}, \mathbf{t}) \leq \mathcal{L}_{\mathrm{P}}(\mathbf{v}, \mathbf{t}), \tag{4}$$

for any $m \geq 1$ and $\mathbf{v} \in \{(v_1, \cdots, v_K) \big| v_k \in \mathbb{R}\}$.

Proposition 1. Suppose class $\mathcal{C}$ consists of data points $\{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{C}|}\}$ associated with a pre-set target $\mathbf{t} \in \mathcal{H}$ in Hamming space. The averaged *intra-class* pairwise Hamming distances among the corresponding binary codes $\{\mathbf{b}_1, \cdots, \mathbf{b}_{|\mathcal{C}|} | \mathbf{b}_i = \mathbf{\Phi}(\mathbf{x}_i; \mathbf{w})\}$ is upper bounded by,

$$\frac{1}{|\mathcal{C}|^2} \cdot \sum_{1 \leq i,j \leq |\mathcal{C}|} \mathcal{D}_h(\mathbf{b}_i, \mathbf{b}_j) \leq \frac{2}{|\mathcal{C}|} \cdot \sum_{1 \leq i \leq |\mathcal{C}|} \mathcal{L}_{\mathrm{P}}(\mathbf{v}_i, \mathbf{t}). \tag{5}$$

Proposition 2. Suppose there are L classes in the dataset, i.e. $\mathcal{C}_1, \cdots, \mathcal{C}_L$. For any two classes $\mathcal{C}_x$ and $\mathcal{C}_y$ $(1 \leq x \neq y \leq L)$, respectively, with associated targets binary vectors $\mathbf{t}_x$ and $\mathbf{t}_y$ and binary hash codes $\mathbf{b}_i^x = \mathbf{\Phi}(\mathbf{x}_i; \mathbf{w}), i \in \{1, \cdots, |\mathcal{C}_x|\}$, $\mathbf{b}_i^y = \mathbf{\Phi}(\mathbf{y}_j; \mathbf{w}), j \in \{1, \cdots, |\mathcal{C}_y|\}$, the averaged *inter-class* pairwise Hamming distances among binary codes $\sum_{\substack{1 \leq i \leq |\mathcal{C}_x|, \\ 1 \leq j \leq |\mathcal{C}_y|}} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^y)$ is lower bounded by,

$$\sum_{1 \leq x \neq y \leq L} \Big(\mathcal{D}_h(\mathbf{t}_x, \mathbf{t}_y) - \frac{1}{|\mathcal{C}_x| \cdot |\mathcal{C}_y|} \cdot \sum_{\substack{1 \leq i \leq |\mathcal{C}_x|, \\ 1 \leq j \leq |\mathcal{C}_y|}} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^y) \Big)$$

$$\leq \sum_{1 \leq x \leq L} \frac{2 \cdot (L-1)}{|\mathcal{C}_x|} \cdot \sum_{1 \leq i \leq |\mathcal{C}_x|} \mathcal{L}_{\mathrm{P}}(\mathbf{v}_i^x, \mathbf{t}_x). \tag{6}$$

Proposition 3. The *difference* between averaged *intra-class* pairwise Hamming distance and averaged *inter-class* pairwise Hamming distance is upper bounded, i.e.

$$\sum_{1 \leq x \leq L} \frac{1}{|\mathcal{C}_x|^2} \cdot \sum_{1 \leq i,j \leq |\mathcal{C}_x|} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^x)$$

$$- \sum_{1 \leq x \neq y \leq L} \frac{1}{|\mathcal{C}_x| \cdot |\mathcal{C}_y|} \sum_{\substack{1 \leq i \leq |\mathcal{C}_x|, \\ 1 \leq j \leq |\mathcal{C}_y|}} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^y)$$

$$\leq \sum_{1 \leq x \leq L} \frac{2 \cdot L}{|\mathcal{C}_x|} \cdot \sum_{1 \leq i \leq |\mathcal{C}_x|} \mathcal{L}_{\mathrm{P}}(\mathbf{v}_i^x, \mathbf{t}_x) - \sum_{1 \leq x \neq y \leq L} \mathcal{D}_h(\mathbf{t}_x, \mathbf{t}_y). \tag{7}$$

Remarks:

I Inequality in (Eq. 5) shows that the averaged polarization loss is a strict upper-bound of the averaged pairwise Hamming distances between points of the same class. That is to say, minimizing the RHS of (Eq. 5) effectively *minimizes* the averaged *intra-class* pairwise Hamming distances.

II In terms of the computational complexity, pairwise Hamming distances on the LHS of (Eq. 5) is $O(|\mathcal{C}|^2)$ while the polarization loss on the RHS of (Eq. 5) is $O(|\mathcal{C}|)$ only.

III Inequality in (Eq. 6) shows that minimizing polarization losses on the RHS of (Eq. 6) effectively *maximizes* the averaged *inter-class* pair-wised Hamming distances on LHS.

IV According to Proposition 3, the optimization problem of simultaneous minimizing the intra-class and maximizing inter-class Hamming distances, i.e.

$$\min_{\mathbf{w}} \sum_{1 \leq x \leq L} \frac{1}{|\mathcal{C}_x|^2} \cdot \sum_{1 \leq i,j \leq |\mathcal{C}_x|} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^x)$$

$$- \sum_{1 \leq x \neq y \leq L} \frac{1}{|\mathcal{C}_x| \cdot |\mathcal{C}_y|} \sum_{\substack{1 \leq i \leq |\mathcal{C}_x|, \\ 1 \leq j \leq |\mathcal{C}_y|}} \mathcal{D}_h(\mathbf{b}_i^x, \mathbf{b}_j^y),$$

is equivalent to the problem of minimizing the averaged polarization loss over the whole data set, i.e.[1]

$$\min_{\mathbf{w}} \sum_{1 \leq x \leq L} \frac{1}{|\mathcal{C}_x|} \cdot \sum_{1 \leq i \leq |\mathcal{C}_x|} \mathcal{L}_{\mathrm{P}}(\mathbf{v}_i^x, \mathbf{t}_x). \tag{8}$$

2.3 Empirical Investigations of Error Terms

LHS of (Eq. 7) is the averaged intra-class Hamming Distance minus the inter-class Hamming distance. This quantity has been used to minimize Hamming distances between similar pairs while maximizing Hamming distances between dissimilar pairs, for both pairwise and triplet based losses e.g. [Norouzi and Blei, 2011; Norouzi *et al.*, 2012]. Proposition 3 proves that LHS is strictly bounded by RHS of (Eq. 7), which is the polarization loss minus the pairwise Hamming distance between the target vectors.

Empirically, Figure 2 shows that the proposed polarization loss is closely related to the intra-class and inter-class Hamming distances between similar/dissimilar data pairs, up to

[1] Due to the limited space, proofs of propositions are omitted from the paper and will be included in an elaborated report.

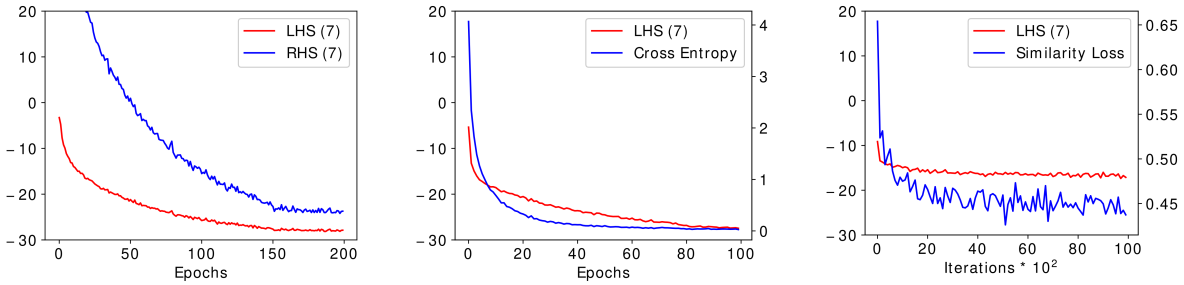

Figure 2: Comparison of minimized loss and target loss i.e. LHS of (Eq. 7) for different methods. (Left): LHS and RHS of (Eq. 7) in Proposition 3, plotted over different epochs of learning. *Pearson correlation* between these two plots is 0.9935. (Middle): Error term in GreedyHash [Su *et al.*, 2018] vs LHS of (Eq. 7). *Pearson correlation* between these two plots is 0.913. (Right): Error term in HashNet [Cao *et al.*, 2017] vs LHS of (Eq. 7). *Pearson correlation* between these two plots is 0.882.

a constant that depends on the pairwise Hamming distances between target vectors. Indeed the measured *Pearson correlation* between these two plots is as high as 0.9935. We view this strong correlation as an unique and favorable feature of the proposed polarization loss. In contrast, error terms used in [Su *et al.*, 2018] and [Cao *et al.*, 2017] neither bounded intra-class nor inter-class Hamming distances. Note that the scales for these error terms are significantly different and the corresponding *Pearson correlation* are measured at 0.913 and 0.882 respectively.

2.4 The Setting of Target Vectors

The minimization of (Eq. 8) is a simple problem in terms of the number of parameters to be optimized — target binary vectors $\mathbf{t}_x$ are constants while network parameters $\mathbf{w}$ for real-valued outputs $\mathbf{v}_i^x(\mathbf{x};\mathbf{w})$ are directly optimized using the back-propagation, thanks to the differentiable polarization loss $\mathcal{L}_{\text{P}}$.

Two strategies have been investigated in the setting of target vectors: (i) random assignments; (ii) adaptive updating with random initialization. Algorithm 1 summarizes the pseudo-codes of both setting schemes.

Algorithm 1 Training of Deep Polarized Network

Input
  x,**t** sample data and target binary vectors
Output
  $\Psi(;\mathbf{w})$trained DPN
procedure TRAIN(Ψ)
  for number of epochs **do**
    $\mathbf{v} \leftarrow \Psi(\mathbf{x};\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} - lr * \nabla_{\mathbf{w}}(\mathcal{L}_p(\mathbf{v},\mathbf{t}))$
    if adaptive updating **then**
      $v^{correct} \leftarrow$ find v with correct prediction
      for each class, c **do**
        $\mathbf{b} \leftarrow \texttt{sign}(v_c^{correct})$
        $\mathbf{t}_c \leftarrow \texttt{sign}(mean(\mathbf{b}, axis = batch))$
      end for
    end if
  end for
end procedure

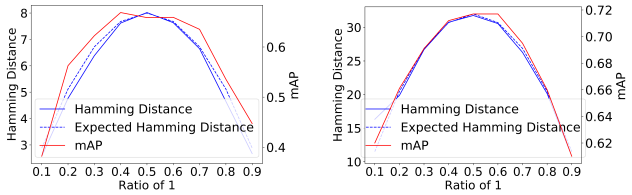

Figure 3: The plot shows the retrieval performance (*mAP*), *measured* and *expected* inter-class hamming distances (see Claim for random assignments of target vectors), with respect to different ratio of positive bits (0.1 up to 0.9). Left to right: ImageNet100 with 16 and 64-bit hash codes respectively.

Random Assignments of Target Vectors

Random projections have long been adopted in the locality sensitive hashing (LSH) as data-independent hashing functions with a theoretical guarantee of successful approximate nearest-neighbour search [Indyk and Motwani, 1998; Andoni and Indyk, 2006]. Following this principle, we adopt a *Random Assignment* scheme for setting target vectors associated with each class. This random assignment scheme is guaranteed to generate target vectors that are of sufficient inter-class hamming distances as shown below.

Claim: Suppose K-bit Hamming space $\mathbb{H}^K := \{-1, 1\}^K$, two binary target vectors $\mathbf{t}_x, \mathbf{t}_y \in \mathbb{H}^K$ are sampled with probability p for 1 on each bit. The expectation of Hamming distance is then given by $E_p(D_h(\mathbf{t}_x, \mathbf{t}_y)) = 2K \cdot p(1-p)$, whereas the expectation achieves maxima K when $p = 0.5$.

Figure 3 plots the expected hamming distances, the measured hamming distances and the hashing accuracies with respect to different ratios of positive bits. Clearly, the maximal inter-class hamming distance gives rise to the optimal hashing accuracy, which is in line with the bound in (Eq. 7). Following the theoretical analysis and the ablation study, we set the ratio of random assignments of positive bits as 0.5 in all the experiments.

Adaptive Updating of Target Vectors

For the adaptive scheme, target vectors are repeatedly updated during the learning of hash functions (see Algorithm 1). It was observed that, (i) frequent updating is not neces-

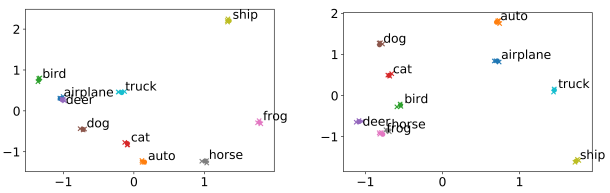

Figure 4: Visualization of the target vectors for each class with our proposed method, projected along first two eigenvectors. (Left): randomly assigned target vectors (i.e. DPN), (Right): adaptively updated target vectors (i.e. DPN-A).

sary or sometimes bring about adverse effect to the convergence of learning process; (ii) the improvements in hashing accuracies brought by the adaptive updating is marginal (see results in Section 3.3); (iii) updated vectors do reflect the intrinsic similarities among different classes e.g. *dog* and *cat* for CIFAR10 dataset (see Figure 4).

3 Experiment Results

This section illustrates the experiment results and ablation study of the hyper-parameter i.e. margin m. Also, an effective ternary assignment is demonstrated with hashing accuracies levitated above the state-of-the-art solutions.

3.1 Datasets and Experiment Settings

CIFAR10 [Krizhevsky, 2009] consists of 10 classes with 60K images. We follow the settings from [Cao *et al.*, 2017] where 100 images per class are selected randomly as query set while the remaining images are used as retrieval set. We randomly sample 500 images per class from the retrieval set for training.

NUS-WIDE [Chua *et al.*, 2009] consists of 81 concepts with 269K multi-labeled images. We follow the settings from [Cao *et al.*, 2017], where 21 of the most frequent concepts are being used as image annotations, and hence 195K images are selected. We selected 100 images per concept randomly as query set while the remaining images as retrieval set. Then we randomly sample 500 images per concept from the retrieval set for training.

ImageNet100 is a subset of ImageNet [Russakovsky *et al.*, 2015] with only 100 classes. We follow the settings from [Cao *et al.*, 2017] and randomly select 100 classes. All the validation images from 100 classes are used as query set while 13K images are randomly sampled from the retrieval set which consists of 128K images.

Our approach is implemented in PyTorch [Paszke *et al.*, 2019], and we used pretrained AlexNet [Krizhevsky *et al.*, 2012] from PyTorch as the initialization and replaced the classification fully-connected layer with a hashing layer consisting of K channels to represent K bits.

3.2 Ablation Study of Margin m

The margin is pre-set $m \geq 1$ for Lemma 1 to hold. As shown in Figure 5, the margin essentially determines the magnitudes of polarization between positive and negative outputs i.e. large margins push the outputs $\mathbf{v}$ further away from zero, and vice versa.

Margin m	CIFAR10	NUS-WIDE	ImageNet100
0.1	0.642	0.785	0.587
0.5	**0.816**	0.836	0.721
1.0	0.812	**0.839**	**0.729**
2.0	0.799	0.838	0.722
3.0	0.805	0.838	0.717

Table 2: Hashing accuracies in terms of mAP with different settings of margin m.

Specifically, $m = 1$ results in the bit-wise hinge loss i.e. $\max(1 - v_i \cdot t_i, 0)$. This form of loss function has long been studied in Support Vector Machine (SVM) for binary classification, as well as has been applied to deep neural networks to replace the cross-entropy loss [Tang, 2013]. Nevertheless, the hinge loss was defined over the dot product of network weights and input features, instead of channel wise outputs in our case. Empirically, we found that $m = 1$ does give rise to the most accurate results while other values of m are sub-optimal (see Table 2). Unless stated otherwise, $m = 1$ is used for all experiments illustrated below.

3.3 Hashing

Hashing accuracies measured by mAP (mean average precision), for different methods are summarized in Table 3. Specifically, four existing methods i.e. HashNet [Cao *et al.*, 2017], MIHash [Cakir *et al.*, 2019], GreedyHash [Su *et al.*, 2018] and JMLH [Shen *et al.*, 2019] are compared with four variants of DPNs. **DPN** denotes the basic deep polarization network with random assignment of target vectors, and **DPN-A** denotes DPN with adaptive updating of target vectors (see Section 2.4). **DPN-T** denotes DPN outputs processed with ternary assignment illustrated in Section 3.3 and **DPN-A-T** is the combinatorial of both aforementioned variants.

It was observed that the hash accuracies achieved by **DPN** is comparable with the-state-of-the-art, e.g. on ImageNet100 datasets, achieving 0.729 and 0.732 with 64 and 128-bits hash codes, respectively. **DPN** also slightly outperformed the best performed JMLH on the multi-labeled NUS-Wide dataset, with 1.9% improvement for 64-bits. For 16-bits, however, **DPN** performances slightly deteriorate, due to the decreased of averaged Hamming distances between target vectors as indicated by (Eq. 7) in Section 2.2.

Although adaptive updating of target vectors reflect intrinsic similarities between classes, as shown in Figure 4, its performances do not consistently improve the random assignment results. Moreover, the adaptive updating actually lead to unstable results for multi-labeled NUS-Wide images. Exploring more stable and effective adaptive updating strategy is one of our future research.

Ternary Assignment

As shown in Figure 5, wrongly retrieved samples often lead to non-polarized outputs. This observation inspired us to adopt an effective ternary assignment as follows:

$$\ddot{\mathbf{\Phi}}(\mathbf{x}; \mathbf{w}) = \begin{cases} -1, & \text{if } x \leq -m \\ 0, & \text{if } -m < x \leq m, \\ 1, & \text{otherwise, for each element } x. \end{cases} \quad (9)$$

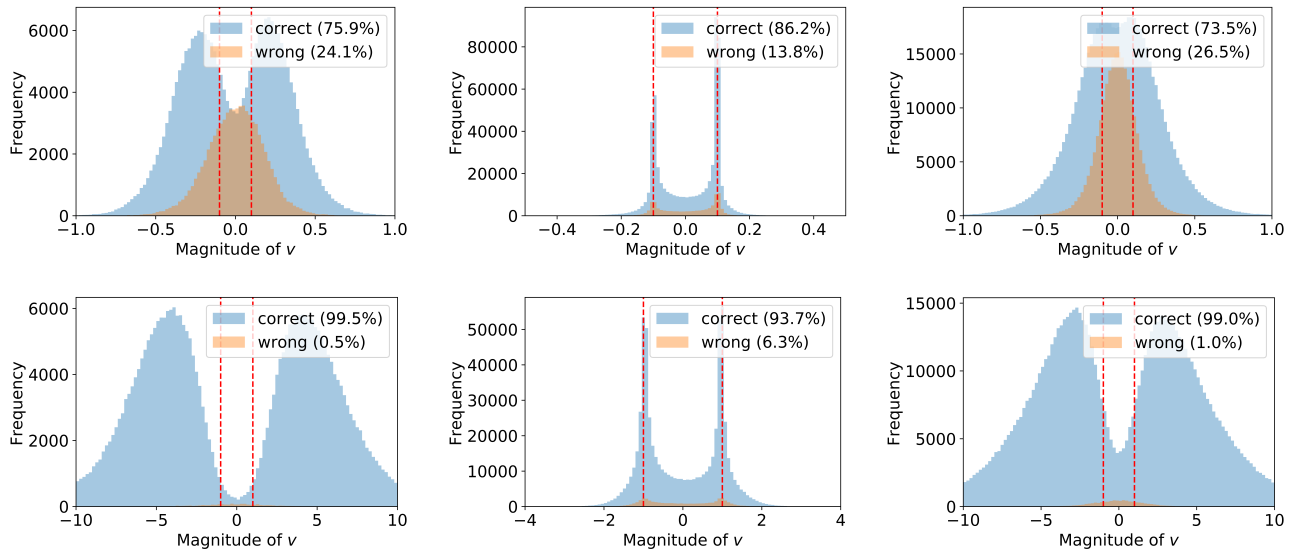

Figure 5: Distributions of the DPN outputs $\mathbf{v}$. Blue distributions represent correctly retrieved samples while orange distributions represent wrongly retrieved samples. Left to right column: Image samples in CIFAR10, NUS-WIDE and ImageNet100 datasets, respectively. Top to bottom row: Different values of margin m (0.1 and 1.0 respectively), which are highlighted with the red dotted line.

Method	**CIFAR10** (mAP@all)				**NUS-WIDE** (mAP@5K)				**ImageNet100** (mAP@1K)			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
DCH [Cao *et al.*, 2018]	-	0.665*	0.673*	0.638*	-	-	-	-	-	-	-	-
HashNet [Cao *et al.*, 2017]	0.643	0.675	0.687	-	0.662	0.699	0.716	-	0.506	0.631	0.684	-
MIHash [Cakir *et al.*, 2019]	0.760	0.776	0.761	-	0.722	0.759	0.779	-	0.569	0.661	0.694	-
DTQ [Liu *et al.*, 2018]	0.789	0.792	0.789*	0.785*	0.798	0.801	-	-	-	-	-	-
GreedyHash [Su *et al.*, 2018]	0.786	0.810	0.814*	0.807*	-	-	-	-	0.625	0.662	0.688	0.699*
JMLH [Shen *et al.*, 2019]	0.805	**0.841**	**0.837**	-	0.795	0.818	0.820	-	0.668	0.714	0.727	-
DPN	0.774	0.803	0.812	0.808	0.810	0.832	0.839	0.840	0.606	0.693	0.729	0.732
DPN-A	0.801	0.821	0.814	0.813	-	-	-	-	0.606	0.687	0.722	0.727
DPN-T	0.789	0.818	0.829	0.823	**0.847**	**0.859**	**0.863**	**0.862**	**0.684**	**0.740**	**0.756**	**0.756**
DPN-A-T	**0.825**	0.838	0.830	**0.829**	-	-	-	-	0.675	0.734	0.751	0.755

Table 3: Hashing accuracies in terms of Mean average precision (mAP) for different methods. Results with * indicate that we run the code released from the authors with default hyperparameters.

Hamming distance between a K-bits ternary code $\ddot{\mathbf{b}}$ and binary code $\mathbf{t}$ is defined as $\mathcal{D}_h(\ddot{\mathbf{b}}, \mathbf{t}) := \frac{1}{2}(K - \ddot{\mathbf{b}} \cdot \mathbf{t})$.

After applying this minor modification to our network outputs, the resulting accuracies of **DPN-T** are consistently levitated above the-state-of-the-art, by elevations more than 2% for both ImageNet100 and NUS-wide datasets. While for CIFAR10, **DPN-T** compared favorably with respect to GreedyHash but not as good as that of JMLH[2]. Finally, the adaptive updating for **DPN-A-T** does not bring about consistent improvements as compared to **DPN-T**.

4 Conclusions

We proposed a novel learning to hash deep neural network, which uses a differentiable hinge-like loss to enforce polarization of each output channel. Contributions of the proposed DPN are three-folds. On the theoretical side, we proved that the proposed polarization loss is intrinsically related to the intra-class and inter-class Hamming distances between similar/dissimilar pairs. We advocate the polarization loss due to its simplicity and differentiability, which allows direct back-propagation to optimize the network weights. Algorithm-wise, a simple *random assignment* strategy for each class of target binary codes turns out to be surprisingly effective and it achieves the state-of-the-art performances. Finally, a ternary assignment is proposed to consistently levitate hashing accuracies by improvements more than 2% for most cases.

Applying channel-wise hinge-like loss is proved to be a simple and effective strategy for hashing in this paper. As one future research direction, we are exploring an effective adaptive updating of target vectors. Other topics within this deep polarization network (DPN) framework, e.g. ternary assignments, call for more research efforts from colleagues interested in DPNs.

[2]We are unable to reproduce the results of JMLH.

Acknowledgements

This work is partially supported by the National Key Research and Development Program of China 208AAA0101100.

References

[Andoni and Indyk, 2006] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[Cakir *et al.*, 2018] F. Cakir, K. He, and S. Sclaroff. Hashing with binary matrix pursuit. In *ECCV*, pages 344–361, 2018.

[Cakir *et al.*, 2019] Fatih Cakir, Kun He, Sarah Bargal, and Stan Sclaroff. Hashing with mutual information. *T-PAMI*, 41(10):2424–2437, 2019.

[Cao *et al.*, 2017] Z. Cao, M. Long, J. Wang, and P. Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, 2017.

[Cao *et al.*, 2018] Y. Cao, M. Long, L. Bin, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *CVPR*, 2018.

[Chua *et al.*, 2009] 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 *CIVR*, 2009.

[Fan *et al.*, 2019] L. Fan, K. W. Ng, and C. S. Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *NeurIPS*, pages 4716–4725, 2019.

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

[He *et al.*, 2018] K. He, F. Cakir, S. A. Bargal, and S. Sclaroff. Hashing as tie-aware learning to rank. In *CVPR*, pages 4023–4032, 2018.

[Hu *et al.*, 2018] Q. Hu, P. Wang, and J. Cheng. From hashing to cnns: Training binary weight networks via hashing. In *AAAI*, 2018.

[Indyk and Motwani, 1998] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM-STOC*, pages 604–613, 1998.

[Jeong and Song, 2018] Y. Jeong and H. O. Song. Efficient end-to-end learning for quantizable representations. In *ICML*, 2018.

[Krizhevsky *et al.*, 2012] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

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

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

[Li *et al.*, 2016] W-J Li, S. Wang, and W-C Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016.

[Lin *et al.*, 2013] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, 2013.

[Lin *et al.*, 2015] K. Lin, H-F Yang, J-H Hsiao, and C-S Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPR-W*, 2015.

[Lin *et al.*, 2017] G. Lin, F. Liu, C. Shen, J. Wu, and H. T. Shen. Structured learning of binary codes with column generation for optimizing ranking measures. *Int. J. Comput. Vision*, 123(2):287–308, June 2017.

[Liu *et al.*, 2012] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

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

[Liu *et al.*, 2018] B. Liu, Y. Cao, M. Long, J. Wang, and J. Wang. Deep triplet quantization. In *ACM-MM*, pages 755–763, 2018.

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

[Norouzi *et al.*, 2012] M. Norouzi, D. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, 2012.

[Paszke *et al.*, 2019] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv e-prints*, page arXiv:1912.01703, Dec 2019.

[Russakovsky *et al.*, 2015] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision*, 115(3):211–252, 2015.

[Shen *et al.*, 2015] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, 2015.

[Shen *et al.*, 2019] Y. Shen, J. Qin, J. Chen, L. Liu, and F. Zhu. Embarrassingly Simple Binary Representation Learning. *arXiv e-prints*, page arXiv:1908.09573, Aug 2019.

[Su *et al.*, 2018] S. Su, C. Zhang, K. Han, and Y. Tian. Greedy hash: Towards fast optimization for accurate hash coding in CNN. In *NeurIPS*, pages 806–815, 2018.

[Tang, 2013] Y. Tang. Deep Learning using Linear Support Vector Machines. *arXiv e-prints*, page arXiv:1306.0239, Jun 2013.

[Wang *et al.*, 2013] J. Wang, W. Liu, A. X. Sun, and Y. Jiang. Learning hash codes with listwise supervision. In *ICCV*, pages 3032–3039, 2013.

[Wang *et al.*, 2015] Q. Wang, Z. Zhang, and S. Luo. Ranking preserving hashing for fast similarity search. In *IJCAI*, pages 3911–3917, 2015.

[Wang *et al.*, 2016] X. Wang, Y. Shi, and K. Kitani. Deep supervised hashing with triplet labels. In *ACCV*, pages 70–84, 2016.

[Wang *et al.*, 2018] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *T-PAMI*, 40(4):769–790, 2018.

[Yang *et al.*, 2018] H. Yang, K. Lin, and C. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *T-PAMI*, 40(2):437–451, 2018.

[Yao *et al.*, 2016] T. Yao, F. Long, T. Mei, and Y. Rui. Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*, 2016.

[Zhao *et al.*, 2015] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[Zhuang *et al.*, 2016] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *CVPR*, 2016.