Classification-Based Supervised Hashing with Complementary Networks for Image Search

Dong-ju Jeong
jeongdj@ispl.snu.ac.kr
Sungkwon Choo
chewry@ispl.snu.ac.kr
Wonkyo Seo
cusisi@ispl.snu.ac.kr
Nam Ik Cho
nicho@snu.ac.kr

Department of Electrical and Computer Engineering, INMC
Seoul National University
Seoul, Republic of Korea

Abstract

For realizing an efficient image search system, the binary hash code representation of images is essential to reduce memory consumption and computation time. In this paper, we present a new supervised hashing method using complementary networks. We construct the framework that performs classification tasks for discriminative hash codes and regularizes them with the assistance of adversarial networks. Specifically, our framework has an encoder network that generates relaxed hash codes and is trained with a classifier to make the codes discriminative. A discriminator network takes as inputs the instances from a desired distribution and generated hash codes to learn to distinguish true and false samples. The classifier presents resulting one-hot labels to the desired instances, which complements the main classification task. As simultaneously trained with the discriminator, the encoder network produces the hash codes with the desired properties. Experiments on widely used datasets show that the proposed framework can help to produce effective hash codes for image search, and applies to hashing with both single- and multi-label classification tasks.

1 Introduction

As one of the important visual recognition topics, content-based image retrieval [38] tackles the problem to transform images into their visual descriptors and search the images relevant to queries. To generate the visual descriptors, many studies proposed the algorithms for necessary procedures such as feature extraction, embedding, and aggregation [17, 19, 21]. However, image search with these real-valued descriptors would bring about heavy memory consumption and computation time, which makes a visual search system less practical. Thus, it is necessary to highly compress the visual descriptors while leaving them visually or semantically meaningful. This is of much more importance in the large-scale visual search scenarios. To design a practical image retrieval system, one would need not only meaningful

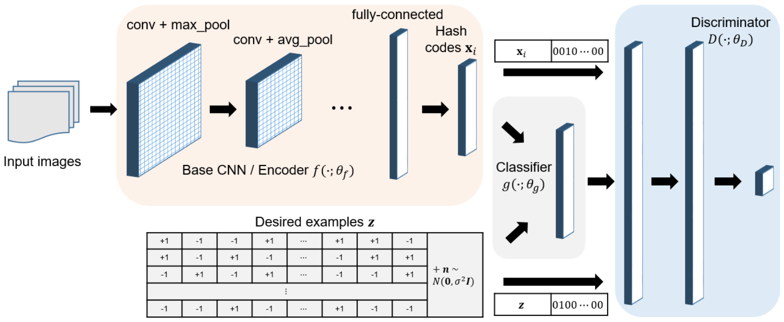

Input images
conv + max_pool
conv + avg_pool
fully-connected
Hash codes $\mathbf{x}_i$
Base CNN / Encoder $f(\cdot;\theta_f)$
$\mathbf{x}_i$ 0010⋯00
Classifier $g(\cdot;\theta_g)$
Discriminator $D(\cdot;\theta_D)$
Desired examples $\mathbf{z}$
$+\mathbf{n} \sim N(\mathbf{0},\sigma^2\mathbf{I})$
$\mathbf{z}$ 0100⋯00

Figure 1: The structure of our proposed model consisting of its encoder, classifier, and discriminator networks. The encoder network produces the hash codes $\mathbf{x}_i$, and the discriminator tells the desired examples $\mathbf{z}$ from the hash codes $\mathbf{x}_i$ with the assistance of the classifier. $\mathbf{z}$ vectors are randomly sampled from $\{-1,+1\}^B$ and combined with their classification results, while $\mathbf{x}_i$ vectors are concatenated to their true labels.

descriptors but also a hashing method with a fast search algorithm [1]. In this paper, we focus on devising a hashing method to turn images into binary hash codes to boost searching efficiency.

In contrast to the data-independent algorithms such as locality sensitive hashing (LSH) [8], the data-dependent methods utilize the structure of data to produce similarity-preserving hash codes. For example, ITQ(-CCA) [10], KSH [32], MLH [36], SSH [46], PQ [18, 19], and SDH [13, 42] optimize their hashing functions in (semi-)supervised or unsupervised manners. These methods transform image descriptors into their hash codes, which can also complement the image retrieval systems based on deep convolutional neural networks (CNNs) [12, 20, 35]. In addition, CNN-based hashing methods have been proposed to receive raw images as inputs, and learn image features and optimal hash codes simultaneously. In [29], an additional hash layer is inserted between fully-connected layers of a deep CNN model, and it constrains input values into the range of $[0,1]$ with sigmoid activation. This hash layer learns relaxed hash codes with the classification of labeled data, which are binarized with simple thresholding. The methods of [26, 28, 30] utilize siamese/triplet networks so that this directly tightens the distances between the hash codes of similar images, whereas those of dissimilar images become more dispersed. In particular, a loss function is added to trigger the absolute value of each bit to get close to 1 in [30]. This loss function is also used for unsupervised hashing in [7]. Likewise, the method of [16] uses additional entropy-based loss functions to produce the concatenations of one-hot codes.

Meanwhile, deep generative models have recently been proposed to exploit the useful frameworks such as generative adversarial networks (GANs) [11]. GAN-based models are used not only to generate realistic samples but also to treat image-to-image translation tasks [15, 49] and other computer vision problems such as semi-supervised classification [41], super-resolution [27], and image retrieval [39, 43]. Moreover, this framework can also be utilized to lead certain latent variables to follow a specific distribution, which is used to train generative models and learn the side information of images [33]. It is notable that generative model frameworks can facilitate the tasks where CNNs play important roles, and they can also be applied to the hashing for image search. In [39, 43], the adversarial networks generate synthetic images, which are used to minimize multiple loss functions for image search. In this work, we focus on treating the distribution of hash codes with such adversarial framework rather than generating the training images.

Considering the issues stated above, we propose a supervised hashing method using complementary networks. In the proposed method, semantically meaningful descriptors can be learned with the assistance of a classification framework. Given the trained classifier and the supervision with labeled data, adversarial networks can help to regularize the useful descriptors to obtain relaxed hash codes. In this situation, an encoder CNN acts as a generative model adversarially trained with a discriminator network. The instances from a specific desired distribution are input to the discriminator as "true" samples, and then the encoder CNN learns to produce the hash codes that are desired from the perspective of the discriminator. Furthermore, the label information and classification results can be utilized to preserve the discriminability of the hash codes. During the training phase, the ground-truth labels of images and the classification results of desired samples are combined with the hash codes and desired samples respectively, as shown in Fig. 1. When the inputs to the discriminator are modified in such way, it can be complementary to the classifier. Trained together with the discriminator, the encoder network learns to turn input images into similarity-preserving codes following the user-defined distribution. Our experiments show that the proposed method well preserves the similarity relationships of the visual descriptors and performs the image search task with effective binary codes.

2 Approach

2.1 Formulation

Given a set of training images $\{I_i, \mathbf{c}_i\}_{i=1}^N$, where N is the number of training data, our goal is to obtain relaxed hash codes $\{\mathbf{x}_i\}_{i=1}^N$ and transform them into binary hash codes $\{\widetilde{\mathbf{x}}_i\}_{i=1}^N$. I_i and $\mathbf{c}_i \in \{0,1\}^L$ are the i-th training image and its L-class one-hot label vector respectively. If we let $f(\cdot;\theta_f)$ denote a function transforming an image into a relaxed hash code with a base CNN and its parameters θ_f, we can say $\mathbf{x}_i = f(I_i;\theta_f) \in \mathbb{R}^B$ and $\widetilde{\mathbf{x}}_i = \text{sign}(\mathbf{x}_i) \in \{-1,+1\}^B$, where B is the code length of $\widetilde{\mathbf{x}}_i$ and $\text{sign}(\cdot)$ is the element-wise sign function that outputs $+1$ if an element is nonnegative and -1 otherwise. We treat the Hamming space $\{-1,+1\}^B$ rather than $\{0,1\}^B$ because the Hamming distance between $\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j \in \{-1,+1\}^B$ corresponds to their Euclidean distance in $\mathbb{R}^B$ [32], so this helps to optimize the relaxed hash codes instead of strictly binary codes.

As in [13, 16, 29, 31, 42], discriminative hash codes can be optimized with the assistance of classifiers, which does not need pairwise labels that are necessary for siamese or triplet networks [26, 30]. Given a classifier $g(\cdot;\theta_g)$ with its parameters θ_g, implemented as a fully-connected layer combined with its activation function, we obtain the probabilistic classification result of an image $\mathbf{y}_i = g(\mathbf{x}_i;\theta_g) \in \mathbb{R}^L$. This is used to optimize the discriminative relaxed hash codes $\mathbf{x}_i$.[1] With the defined functions and variables, an optimization problem can be set for the supervised hashing as below:

$$\min_{\theta_f,\theta_g} \sum_{i=1}^{N} \left(L_{\text{cls}}(\mathbf{y}_i, \mathbf{c}_i; \theta_f, \theta_g) + L_{\text{reg}}(\mathbf{x}_i; \theta_f)\right) \quad (1)$$

[1]We use the terms $g(\cdot;\theta_g)$ and $D(\cdot;\theta_D)$ interchangeably with $g(\cdot)$ and $D(\cdot)$ respectively in this paper.

where L_{cls} is the classification loss related to $\mathbf{c}_i$ and $\mathbf{y}_i = g(\mathbf{x}_i; \theta_g)$, and L_{reg} is a regularizer to lead $\mathbf{x}_i$ to follow a desired distribution, considering its discriminability and quantization error when transformed into $\widetilde{\mathbf{x}}_i$. The overall structure of the proposed method is described in Fig. 1.

2.2 Regularization with Adversarial Networks

It is desired that $\mathbf{x}_i$ should be a similarity-preserving and discriminative code and its elements be independent and identically distributed. For this, we define the regularizer L_{reg} and its auxiliary loss function $L_{\text{aux}}(\mathbf{z}, \mathbf{x}_i; \theta_d)$ to constrain $\mathbf{x}_i$ to follow some instances $\mathbf{z}$ sampled from the desired distribution specified below. To define L_{aux}, we construct a discriminator network $D(\cdot; \theta_D)$ that distinguishes desired instances $\mathbf{z}$ from generated hash codes $\mathbf{x}_i$. In this setting, L_{reg} and L_{aux} can be defined, treating the encoder $f(\cdot; \theta_f)$ and the discriminator $D(\cdot)$ as adversarial networks [11]:

$$L'_{\text{reg}} = -\sum_{i=1}^{N} \log\left[D(\mathbf{x}_i)\right], \quad L'_{\text{aux}} = -\sum_{j} \log[D(\mathbf{z}_j)] - \sum_{i=1}^{N} \log\left[1 - D(\mathbf{x}_i)\right] \tag{2}$$

where L'_{reg} and L'_{aux} are the summed loss functions for all the training data, minimized with the encoder $f(\cdot; \theta_f)$ and the discriminator $D(\cdot)$ respectively. In Eq. (2), each negative log-likelihood term can also be seen as a cross-entropy loss with consistent labels $\mathbf{1}$ or $\mathbf{0}$. The loss term for the encoder network L'_{reg} is modified following the proposition of [11] for stronger gradients at the beginning of training.

While many GAN-based generative models receive random noise and some auxiliary information in order to output random generated images [2, 3, 11, 40], other methods for image-to-image translation process input images to obtain pixel-wise resulting maps [15, 49]. The common feature of these methods is to treat "real" training data as inputs to their discriminators with "true" labels. In addition, adversarial networks can also be used to constrain some latent variables to follow a specific distribution [33]. In this case, "real" inputs to the discriminator are samples from the desired distribution, and "fake" ones are generated latent variables. To this end, we need to define the desired distribution from which the instances $\mathbf{z}$ are sampled. Then, the discriminator $D(\cdot)$ and the encoder $f(\cdot, \theta_f)$ are trained to distinguish $\mathbf{z}$ from the generated latent variables $\mathbf{x}_i$ and lead them to follow the desired distribution.

2.3 Desired Distribution of Hash Codes

Basically, we begin with defining $\mathbf{z}$ in the form of $\mathbf{z} = \mathbf{b} + \mathbf{n}$, where $\mathbf{b} \in \{-1, +1\}^B$, $\mathbf{n} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$. If $\mathbf{b}$ is uniformly sampled from $\{-1, +1\}^B$, minimizing the loss functions of Eq. (2) means that $\mathbf{x}_i$ should be similar to any binary codes with the length of B. In order to allow for rooms around $\mathbf{b}$, we add $\mathbf{n}$ to $\mathbf{b}$, which facilitates the adaptation of $\mathbf{x}_i$. This is also related to the observation that excessively fitting relaxed codes to strictly binary ones results in unsatisfactory performance [30]. In other words, this helps the classification and regularization tasks to be compatible with each other, where the former one is for similarity-preserving codes and the latter one for the codes with lower quantization error.

However, using $\mathbf{x}_i$ and $\mathbf{z} = \mathbf{b} + \mathbf{n}$ stated above as the inputs to the discriminator is insufficient to obtain similarity-preserving and discriminative codes, even likely to corrupt these desired properties. Hence, we propose several simple modifications to tackle this problem.

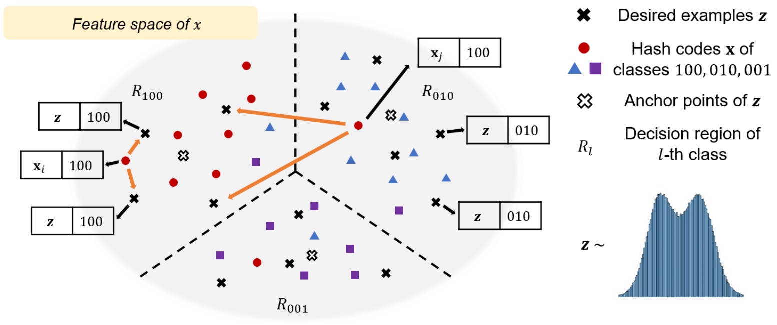


Figure 2: The illustration showing the behaviors of $\mathbf{x}_i$ and $\mathbf{z}$. Red circles are training samples of the class "100" and should approach the desired instances located in the decision region of their class. As training the encoder $f(\cdot,\theta_f)$ with the discriminator, the red circles in R_{100} get similar to the $\mathbf{z}$ vectors in R_{100}, while those in the other decision regions find appropriate positions in the right region R_{100}.

First, we concatenate two kinds of one-hot label vector to $\mathbf{x}_i$ and $\mathbf{z}$. A probability vector $g(\mathbf{z};\theta_g)$ tells us the decision region of which class $\mathbf{z}$ is located in with the *currently* given classifier $g(\cdot;\theta_g)$. If we let $h(\cdot)$ denote the function transforming a probability vector into a one-hot vector, which changes the largest element into 1 and 0s for others, we can use $h(g(\mathbf{z}))$ as an indicator that tells us $\mathbf{z}$ belongs to the l-th class, where $h(g(\mathbf{z}))_l = 1$. In other words, the vectors $\mathbf{z}$ concatenated with one-hot labels of the same class are commonly located in the decision region of that class and are probably close to each other. In this case, we can also divide the set $\{\mathbf{z}\}$ into disjoint subsets $\mathbf{Z}_1,...,\mathbf{Z}_L$, where $\mathbf{Z}_l = \{\mathbf{z}|h(g(\mathbf{z}))_l = 1,\ \mathbf{z}\in\{-1,+1\}^B\}$. On the other hand, to make full use of ground truth labels, these one-hot vectors $\mathbf{c}_i$ are attached to $\mathbf{x}_i$, indicating that $\mathbf{x}_i$ should be located in the decision region of its true class. Thus, we can define the extended versions of the desired instances and generated latent variables:

$$\mathbf{z}^+ = [\mathbf{z}, h(g(\mathbf{z}))],\ \mathbf{x}_i^+ = [\mathbf{x}_i, \mathbf{c}_i] \in \mathbb{R}^{B+L}. \tag{3}$$

If $\mathbf{z}^+$ and $\mathbf{x}_i^+$ are input to the discriminator instead of $\mathbf{z}$ and $\mathbf{x}_i$, it is trained to recognize $\mathbf{z}^+$ as "real" instances and reject "fake" ones $\mathbf{x}_i^+$. Specifically, if a training sample $\mathbf{x}_i$ of the l-th class is not similar to the elements of $\mathbf{Z}_l$, it should obtain a low value of $D(\mathbf{x}_i^+)$. 1) In the case of $\mathbf{c}_i = h(g(\mathbf{x}_i))$, $\mathbf{x}_i$ is forced to be close to the neighboring $\mathbf{z}$ in the decision region of its class. 2) In the case of $\mathbf{c}_i \neq h(g(\mathbf{x}_i))$, the discriminator also leads the encoder network $f(\cdot,\theta_f)$ to locate $\mathbf{x}_i$ in its true decision region as illustrated in Fig. 2. Thus, the optimization with the adversarial networks is also complementary to the classification task with L_{cls}. The adversarially learned inference (ALI) [6] also utilizes two types of concatenated vectors as the inputs to its discriminator. It generates the pairs of real/generated images and inferred/sampled features, while the proposed method can be said to use the ground-truth label vectors and classification results instead of the real and generated images respectively for the discriminative codes.

Moreover, we need to make the hash codes $\mathbf{x}_i$ discriminative, but the support of the desired distribution is too large. Hence, we define an anchor point set $\mathbf{P} = \{\mathbf{p}_1,...,\mathbf{p}_L\}$, where $\mathbf{p}_l \in \{-1,+1\}^B$, and replace $\mathbf{b}\in\{-1,+1\}^B$ with $\widetilde{\mathbf{b}}\in\mathbf{P}$. The anchor points can be set out of randomly generated samples from $\{-1,+1\}^B$ by simply clustering them and finding their cluster centers with k-means clustering. This process in our experiments is repeated many times, and the most dispersed cluster centers are selected. However, the anchor points generated in such way cannot reflect the differences in semantic information and correlation between classes. Thus, we can set these vectors as initial anchor points and update them during the training process. If $\mathbf{p}_l^t$ denotes the l-th anchor point at the t-th step of training, the anchor points can be updated using moving averages as below:

$$\widetilde{\mathbf{p}}_l^{t+1} = \alpha \widetilde{\mathbf{p}}_l^t + (1-\alpha)\frac{1}{N_{T,l}}\sum_{k=1}^{N_T} \mathbb{1}\left[c_{k,l}=1\right]\mathbf{x}_k, \quad \mathbf{p}_l^{t+1} = \mathrm{sign}(\widetilde{\mathbf{p}}_l^{t+1}),\ \widetilde{\mathbf{p}}_l^1 = \mathbf{p}_l^1 \tag{4}$$

where $\mathbf{x}_k$ is the k-th hash code in the current mini-batch of the size N_T, $c_{k,l}$ is the l-th element of $\mathbf{c}_k$, and $N_{T,l} = \sum_{k=1}^{N_T} \mathbb{1}\left[c_{k,l}=1\right]$. The anchor points are initialized out of random samples by the clustering step and are updated with moving averages of hash code samples that belong to each class.

2.4 Integrated Loss Functions

Given the concrete loss functions and variables, the precise form of our optimization problem can be specified as below:

$$\begin{aligned} &\min_{\theta_f,\theta_g} \sum_{i=1}^{N} \left(L_{\mathrm{cls}}(\mathbf{y}_i, \mathbf{c}_i; \theta_f, \theta_g) - \lambda \log\left[D(\mathbf{x}_i^+)\right]\right), \\ &\min_{\theta_D} -\sum_j w_j \log[D(\mathbf{z}_j^+)] - \sum_{i=1}^{N} \log\left[1 - D(\mathbf{x}_i^+)\right],\ w_j = \mathbb{1}\left[\max(g(\mathbf{z}_j)) \geq \tau\right] \end{aligned} \tag{5}$$

where L_{cls} is the cross-entropy loss function of $\mathbf{y}_i$ and $\mathbf{c}_i$, and λ is the controlling parameter that balances the classification loss and the encoder loss. This optimization procedure has two training phases, each of which corresponds to each problem of Eq. (5). Even though one may perform three-phase training, separating the optimization of the classifier and encoder, our model can be stably optimized in the two-phase training partly because our adversarial networks are optimized complementing the classification task. In addition, we discard part of desired samples $\mathbf{z}$ that are close to their decision boundaries using a threshold τ. This helps the discriminator to ignore less discriminative instances from the perspective of the currently given classifier $g(\cdot;\theta_g)$.

3 Experimental Results

3.1 Experimental Setup

In our experiments, two widely used datasets with category labels, CIFAR-10 [23] and NUS-WIDE [4], are used to evaluate the performance of our method and compare it with others. The CIFAR-10 dataset consists of 10 classes, each of which includes equally 6,000 images of the size 32×32, and is divided into 50,000 and 10,000 images for training and testing respectively. The NUS-WIDE dataset is composed of 269,648 Flickr images and each image has its multi-label vector connecting it to part of 81 concepts. For comparison with previous works [26, 30], 21 most frequent concepts are selected, each of which has 5,099 or more images, and contain 195,834 images in total. In the literature, CNN-based models process raw images as their inputs, while input images are transformed into fixed-length descriptors such as GIST [37] for many other methods [10, 32, 42]. Following [30], we define the relevance of test images using the category labels of CIFAR-10 and NUS-WIDE. In particular, as for NUS-WIDE, two images are considered relevant to each other if they have one or more common concepts. To select 10,000 test query images, we repeatedly sample random test query sets and choose the one with the concept labels the distribution of which is most similar to that of the whole dataset. Then, the rest of the database images are used for training.

For each evaluation dataset, the performance is measured with three widely used criteria: the mean average precision (mAP), the precision-recall (PR) curves, and the mean precision scores within several Hamming radii. The code length B is set to one of $\{12, 24, 36, 48\}$ for comparison. The average precision can be calculated with the area under the PR curve, and it is approximated by summing the areas of rectangles never deviating from the curve, which is slightly different from the standard evaluation protocol of [38].

The networks of our proposed model are based on the base CNN commonly used in [16, 30]. This base CNN consists of three 5×5 convolutional layers, which have 32, 32, and 64 channels respectively, and two fully-connected layers with 500 and B units, respectively. As illustrated above, we add a classifier output layer to the base CNN and construct a discriminator network consisting of two fully-connected layers with rectified linear unit (ReLU) activation and batch normalization [14], and one output unit with sigmoid activation. The base CNN behaves as the encoder (or a generator), and this network and discriminator play the roles of adversarial networks. All of our network weights are initialized with the Xavier method [9], and the network is trained by the stochastic gradient descent algorithm with the Adam solver [22]. We set the learning rate to 0.001, and the momentum parameters are set to 0.9 for $f(\cdot, \theta_f)$ and $g(\cdot, \theta_g)$, and 0.1 for $D(\cdot, \theta_D)$. The weight decay is set to 0.004. The last layer of $f(\cdot, \theta_f)$ contains the linear or tanh activation function for CIFAR-10 or NUS-WIDE respectively. The classifier $g(\cdot, \theta_g)$ performs the softmax or sigmoid activation for CIFAR-10 or NUS-WIDE respectively. As for NUS-WIDE, the training images are resized to 64×64, irrespective of their original sizes and aspect ratios. We select NUS-WIDE as an evaluation dataset to confirm the applicability of our proposed method to multi-label classification, so a few modifications are applied to our method. When $\mathbf{c}_i$ is concatenated to $\mathbf{x}_i$, only one out of the several 1s in $\mathbf{c}_i$ is randomly selected and the others are set to zero at each training step. In such way, $\mathbf{x}_i$ is led to get close to decision boundaries of its concepts. Furthermore, w_j in Eq. (5) is not used since we should provide the discriminator with the desired instances $\mathbf{z}$ close to decision boundaries. We consistently set λ and α to 1.0 and 0.999 respectively, where in particular, it was confirmed that the results are not sensitive to the value of λ in our experiments. We use $\tau = 0.3$ (for CIFAR-10) and $\sigma = 0.7$, and also conduct grid search experiments for τ and σ to ensure that we choose the appropriate values of these parameters. In addition, we replace the negative log-likelihood loss functions for the encoder and discriminator with mean squared error terms [34, 49], and smooth the positive labels of the adversarial networks to 0.9 [41] for stable training and better performance.

3.2 Comparison with the State-of-the-Art

With the evaluation criteria stated above, we compare the proposed supervised hashing method (SHAN) with other major algorithms. In the literature, researchers have conducted experiments in different settings, varying in base CNNs, training/query-test data, image size, the number of retrieved data, and so on. Thus, we carefully select the comparable methods whose results can be given in the same settings with few exceptions. They range from the

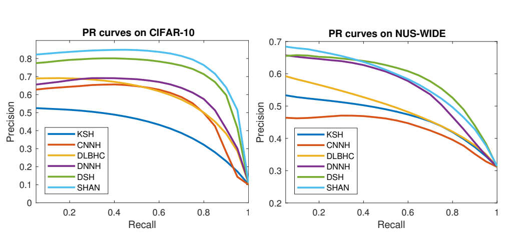
Table 1: Quantitative comparison on the CIFAR-10 and NUS-WIDE datasets with the mean average precision (mAP) measure.

Method	CIFAR-10				NUS-WIDE			
	12 bits	24 bits	36 bits	48 bits	12 bits	24 bits	36 bits	48 bits
LSH [8]	0.1277	0.1367	0.1407	0.1492	0.3329	0.3392	0.3450	0.3474
SH [47]	0.1319	0.1278	0.1364	0.1320	0.3401	0.3374	0.3343	0.3332
BRE [25]	0.1589	0.1632	0.1697	0.1717	0.3556	0.3581	0.3549	0.3592
MLH [36]	0.1844	0.1994	0.2053	0.2094	0.3829	0.3930	0.3959	0.3990
ITQ [10]	0.1080	0.1088	0.1117	0.1184	0.3425	0.3464	0.3522	0.3576
ITQ-CCA [10]	0.1653	0.1960	0.2085	0.2176	0.3874	0.3977	0.4146	0.4188
KSH [32]	0.2948	0.3723	0.4019	0.4167	0.4331	0.4592	0.4659	0.4692
CNNH [48]	0.5425	0.5604	0.5640	0.5574	0.4315	0.4358	0.4451	0.4332
DLBHC [29]	0.5503	0.5803	0.5778	0.5885	0.4663	0.4728	0.4921	0.4916
DNNH [26]	0.5708	0.5875	0.5899	0.5904	0.5471	0.5367	0.5258	0.5248
BDNN [5]	—	0.6521	—	0.6653	—	—	—	—
DSH [30]	0.6157	0.6512	0.6607	0.6755	0.5483	0.5513	0.5582	0.5621
SUBIC [16]	0.6349	0.6719	0.6823	0.6863	—	—	—	—
SHAN	**0.7105**	**0.7556**	**0.7634**	**0.7681**	**0.5660**	**0.5779**	**0.5803**	**0.5829**

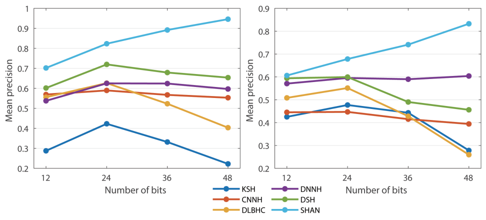
ones that process transformed image descriptors to CNN-based ones that receive input images and encode them into binary codes: LSH, SH, BRE, MLH, ITQ(-CCA), KSH, CNNH, DLBHC, DNNH, BDNN, DSH, and SUBIC. The mAP scores of BDNN are produced with AlexNet [24], but it is deeper than the base CNN used in this work.

First, the quantitative results with mAP values are shown in Table 1, where it can be seen that our proposed method outperforms the other compared ones in image search tasks. Even though the scores of LSH to KSH are given with the GIST descriptors as in many previous works, [28] conducted the experiments with CNN features, which showed that such methods performed worse than the CNN-based ones nonetheless. It is noteworthy that the CNN-based methods vary in the strategies of optimizing relaxed hash codes. For example, some hashing methods such as DLBHC [29] and DNNH [26] design their hash layers with sigmoid or tanh activations and apply thresholding to the real-valued codes to obtain binary hash codes. In addition to this, others such as DSH [30], SUBIC [16], and [44] proposed to create additional loss functions for hash codes to have intended forms (i.e., close to one-hot codes or ones from $\{-1,+1\}^B$) and desired statistical properties, which means that different bits are uncorrelated with each other. However, these loss functions are not complementary to the main classification loss or contrastive loss for siamese networks. In contrast, the proposed regularization losses assist the main classification task, simultaneously leading the hash codes to the desired regions in their space. In our experiments, this aspect is especially effective for the single-label classification with CIFAR-10 as can be seen in Table 1. On the other hand, the DNNH [26] and DSH [30] methods exploit the siamese/triplet networks to produce similarity-preserving hash codes using the contrastive losses, which can be said to be well suited for multi-label datasets such as NUS-WIDE. Even though the proposed method simply performs the multi-label classification task, it shows competitive or better performances compared to the above mentioned methods.

Secondly, Fig. 3(a) shows the PR curves of ours and several compared methods drawn with their 48-bit codes. As can be seen in the PR curve for CIFAR-10, the proposed method results in the highest precision in the whole range of recall. The PR curve for NUS-WIDE indicates that the proposed model is less effective for the multi-label data, but it shows competitive performance compared to the state-of-the-art methods especially in the lower range of recall. This is probably more important in terms of the image search tasks. Fig. 3(b) shows

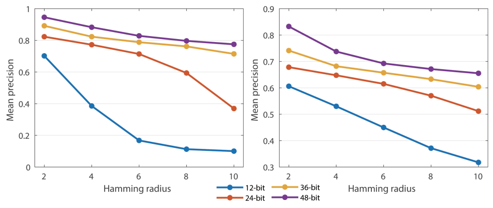


(a)

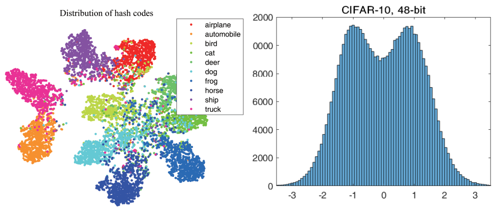


(b)

Figure 3: (a) Quantitative comparison on the CIFAR-10 and NUS-WIDE datasets with the PR curves, which show the precision values corresponding to the recall values in the range of $[0.05, 1.0]$. (b) Mean precision curves within Hamming radius 2 with 48-bit hash codes for CIFAR-10 (left) and NUS-WIDE (right). These curves show the performances of the compared methods in the lower range of recall.



(a)



(b)

Figure 4: (a) Mean precision within several Hamming radii of our method for CIFAR-10 (left) and NUS-WIDE (right). (b) Left: the t-SNE visualization for the 48-bit hash codes of the test query images in CIFAR-10. It is notable that the points of each class are densely located, which helps to improve image search performance. Right: the histogram of the 48-bit relaxed hash codes with $\sigma = 0.7$.

the precision curves within Hamming distance 2, where the proposed method significantly outperforms the others regarding every number of bits in our experiments. This shows the precision for several top-ranked results, which is the proxy for the precision in much lower range of recall that is not shown in Fig. 3(a). In contrast to the other methods, the precision values of our method monotonically increase with the number of bits for both two datasets. This indicates that the test query images relevant to each other robustly agglomerate and semantic differences between images are well reflected to the Hamming distances. The precision values are also related to the quantization errors that occur when relaxed hash codes are quantized to binary ones. If longer codes are allowed, it helps to produce more discriminative hash codes, but at the same time, the quantization errors may be more accumulated along with the bits. Even when more quantization errors occur, the binary codes of our method sustain their neighboring relationships within each class, benefiting from more bits for discriminative codes. Fig. 4(a) shows the mean precision values over several Hamming radii of our method. As expected, the values for shorter codes decrease more rapidly with the Hamming radius, which means that more neighboring test images share the same hash codes or have the ones very similar to each other.

In addition, Fig. 4(b) shows our 48-bit hash codes of the test query set in CIFAR-10. Their dimensionality has been reduced with t-SNE [45] to represent them on a 2D plane.

This shows that the test hash codes agglomerate well, forming the cluster of each class. At the same time, the test hash codes follow the distribution of the desired instances that are input to the discriminator as shown in the histogram of Fig. 4(b).

3.3 Parameter Analysis

The grid search experiments are conducted to find the appropriate values of σ and τ. Table 2 shows the mAP scores along with certain ranges of these parameters. As can be seen in the upper side of Table 2, the most effective values for CIFAR-10 and NUS-WIDE are similar to each other, which are around 0.7, so we can choose the decent values of σ. However, the proposed method produces fine results with the lower values of σ for CIFAR-10, while higher values are suitable for NUS-WIDE. This difference is related to the selection of activation functions at the last layer of $f(\cdot;\theta_f)$. In our experiments for the multi-label classification, we observe that it is difficult to force the values of $\mathbf{x}_i$ to be around ±1 with linear activations. Thus, we apply the tanh activations to NUS-WIDE, and truncate the elements of $\mathbf{z}$ into the range of $[-1,+1]$. On the other hand, the linear activations are applied to CIFAR-10, which produces the distribution of $\mathbf{x}_i$ more similar to that of $\mathbf{z}$, as shown in Fig. 4(b). We also confirm the influences of τ on the mAP results as shown in Table 2. It is notable that the proposed model is not sensitive to the value of τ, so the decent value of τ can be selected. The value of σ used by the proposed method is not large to provide each anchor point with many distant neighbors, so the value of τ does not largely affect the distribution of $\mathbf{z}$. On the other hand, the use of τ for the multi-label classification tends to obstruct the training of our model, so τ is not used for NUS-WIDE.

Table 2: Grid search results (mAP) for σ and τ with CIFAR-10 and NUS-WIDE.

Dataset	σ					τ			
	0.1	0.4	0.7	1.0	1.3	0.1	0.3	0.5	0.7
CIFAR-10	0.758	0.763	0.768	0.698	0.724	0.761	0.768	0.765	0.766
NUS-WIDE	0.509	0.580	0.583	0.576	0.570	—	—	—	—

4 Conclusions

We have proposed a supervised hashing method using complementary networks, which consists of its encoder, classifier, and discriminator networks. This model performs the two-phase training to fit the relaxed hash codes to the specific desired distribution. The encoder network learns to produce effective hash codes for both the classification and adversarial training. The produced hash codes and desired instances are concatenated to ground-truth label vectors and classification results respectively, which are input to the discriminator network as real and fake instances to be distinguished. Experimental results show that the proposed algorithm produces the hash codes that are effective for the image search task, following the desired distribution.

Acknowledgements

This research was financially supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the "Regional Specialized Industry Development Program (R&D, P0002072)" supervised by the Korea Institute for Advancement of Technology (KIAT).

References

[1] A. Babenko and V. Lempitsky. The inverted multi-index. *TPAMI*, 37(6):1247–1260, Jun. 2015.

[2] D. Berthelot, T. Schumm, and L. Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, May 2017.

[3] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180, 2016.

[4] T.-S Chua, Jinhui Tang, R Hong, H Li, Z Luo, and Y Zheng. NUS-WIDE: A real-world web image database from national university of singapore. In *CIVR*, pages 8–10, 2009.

[5] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *ECCV*, pages 219–234, 2016.

[6] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *ICLR*, 2017.

[7] S. En, B. Crémilleux, and F. Jurie. Unsupervised deep hashing with stacked convolutional autoencoders. In *ICIP*, pages 3420–3424, 2017.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[10] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35 (12):2916–2929, Dec. 2013.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[12] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *ECCV*, pages 241–257, 2016.

[13] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan. Fast supervised discrete hashing. *TPAMI*, PP(99):1–1, 2017.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[15] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 5967–5976, 2017.

[16] H. Jain, J. Zepeda, P. Pérez, and R. Gribonval. SUBIC: A supervised, structured binary code for image search. In *ICCV*, pages 833–842, 2017.

[17] H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. In *CVPR*, pages 3310–3317, 2014.

[18] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, Jan. 2011.

[19] H. Jégou, F. Perronnin, M. Douze, J. SÃąnchez, P. PÃľrez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716, Sep. 2012.

[20] D. Jeong, S. Choo, W. Seo, and N. I. Cho. Regional deep feature aggregation for image retrieval. In *ICASSP*, pages 1737–1741, 2017.

[21] Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV*, pages 685–701, 2016.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[23] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.

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

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

[26] H. Lai, Y. Pan, Ye Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[27] C. Ledig, L. Theis, F. HuszÃąr, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 105–114, 2017.

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

[29] K. Lin, H. F. Yang, J. H. Hsiao, and C. S. Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPRW*, pages 27–35, 2015.

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

[31] L. Liu, A. Rahimpour, A. Taalimi, and H. Qi. End-to-end binary representation learning via direct binary embedding. In *ICIP*, pages 1257–1261, 2017.

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

[33] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *ICLR*, 2016.

[34] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv preprint arXiv:1611.04076*, Nov. 2016.

[35] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pages 3456–3465, 2017.

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

[37] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, May 2001.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, pages 1–8, 2007.

[39] Z. Qiu, Y. Pan, T. Yao, and T. Mei. Deep semantic hashing with generative adversarial networks. In *SIGIR*, 2017.

[40] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, Nov. 2015.

[41] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In *NIPS*, pages 2234–2242, 2016.

[42] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[43] J. Song. Binary generative adversarial networks for image retrieval. *arXiv preprint arXiv:1708.04150*, Aug. 2017.

[44] J. Song, T. He, L. Gao, X. Xu, and H. T. Shen. Deep region hashing for efficient large-scale instance search from images. *arXiv preprint arXiv:1701.07901*, Jan. 2017.

[45] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9:2579–2605, Nov. 2008.

[46] J. Wang, S. Kumar, and S. F. Chang. Semi-supervised hashing for large-scale search. *TPAMI*, 34(12):2393–2406, Dec. 2012.

[47] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[48] 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, 2014.

[49] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.