Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Progressive Generative Hashing for Image Retrieval

Yuqing Ma, Yue He, Fan Ding, Sheng Hu, Jun Li, Xianglong Liu*
State Key Lab of Software Development Environment, Beihang University, China
mayuqing@nlsde.buaa.edu.cn, superheyueaaa@buaa.edu.cn, darrow0924@gmail.com,
husheng_7@163.com, junmuzi@gmail.com, xlliu@nlsde.buaa.edu.cn

Abstract

Recent years have witnessed the success of the emerging hashing techniques in large-scale image retrieval. Owing to the great learning capacity, deep hashing has become one of the most promising solutions, and achieved attractive performance in practice. However, without semantic label information, the unsupervised deep hashing still remains an open question. In this paper, we propose a novel progressive generative hashing (PGH) framework to help learn a discriminative hashing network in an unsupervised way. Different from existing studies, it first treats the hash codes as a kind of semantic condition for the similar image generation, and simultaneously feeds the original image and its codes into the generative adversarial networks (GANs). The real images together with the synthetic ones can further help train a discriminative hashing network based on a triplet loss. By iteratively inputting the learnt codes into the hash conditioned GANs, we can progressively enable the hashing network to discover the semantic relations. Extensive experiments on the widely-used image datasets demonstrate that PGH can significantly outperform state-of-the-art unsupervised hashing methods.

1 Introduction

Hashing technique has been widely known for its successful applications to many nearest neighbor search tasks, such as large-scale visual search [He *et al.*, 2012; Liu *et al.*, 2013; Wang *et al.*, 2015], machine learning [Mu *et al.*, 2014], recommendation system [Liu *et al.*, 2014b], etc. It works based on the concept of Locality-Sensitive Hashing (LSH), which guarantees that the nearest neighbors share the similar binary codes, and thus enables fast search with compressed storage over gigantic databases. In the literature, LSH was first introduced by Indyk, and further developed in [Datar *et al.*, 2004], which proposed the popular random projection paradigm. The projection based hashing usually generates the binary codes by first linearly projecting the data along certain directions and then quantizing the projections to binary bits.

Due to the simple form and efficient computation, many following studies have attempted to further improve the discriminative power of the hash codes, and learnt the projection based hash functions by leveraging the information contained in the data [Weiss *et al.*, 2008; Gong and Lazebnik, 2011; Wang *et al.*, 2017; Liu *et al.*, 2014b; Wang *et al.*, 2018; Kong and Li, 2012; Yu *et al.*, 2014; Liu *et al.*, 2014a; Shen *et al.*, 2015; Wang *et al.*, 2014]. Despite the progress in the linear projection based hashing, these methods still cannot well capture the nearest neighbor relations using the binary codes, mainly due to the fact that the shallow structures lack the capability of modelling complex data distribution [He *et al.*, 2013]. Although the nonlinear mapping techniques like kernel can help uplift the data into an informative space [Raginsky and Lazebnik, 2009; Kulis and Darrell, 2009; Liu *et al.*, 2011], they are usually time-consuming, and meanwhile still fail to exploit the underlying data structures using the linear shallow model.

As the deep convolutional neural network yield breakthrough performance on many computer vision tasks, there are several recent studies having been devoted to learning deep hash functions. In these deep network based hashing methods, the hash functions can directly encode the raw images into their binary codes in an end-to-end way. To learn the discriminative hash function, usually we have to train the deep model using a sufficient number of labeled data, which largely limits the power of the deep network based hashing.

There are quite a few deep network based hashing studies that mainly focused on the more generic unsupervised settings, due to the difficulty of training the hashing networks. DH [Liong *et al.*, 2015] and Deepbit [Lin *et al.*, 2016] serve as the pioneering work that learnt the nonlinear end-to-end hash function under independent and balanced constraints on the generated binary codes. Without any supervision information, both methods have shown the encouraging performance in image retrieval. More recently, in [Song, 2017] the more powerful Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014] have been further adopted to generate binary representation, which achieved very promising performance compared to the other unsupervised deep hashing methods, but heavily relies on the pre-defined similarity matrix as the side information.

To exhaustively exploit the power of GAN itself, this paper proposes a novel progressive generative adversarial frame-

*Corresponding Author

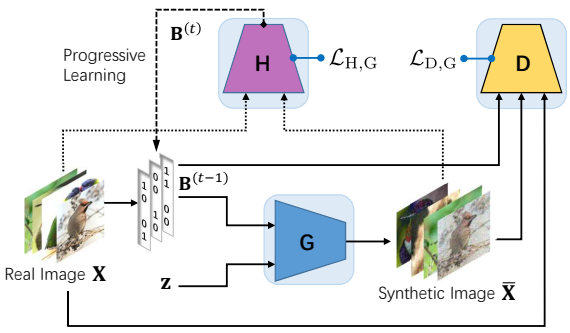


Figure 1: The architecture of our proposed PGH model

work to help learn a discriminative deep hashing network in an unsupervised way. Very different from [Song, 2017], our model employs a progressive way to gradually generate more discriminative hash codes, based on the generated semantic and realistic images by GAN. Specifically, for each image we regard its binary code generated by any weak hashing method as a kind of weak semantic condition for the similar image generation, and simultaneously feed it and the original image into the designed GAN to generate the synthetic image. With the real images and their synthetic ones, we can further train a discriminative hashing network by minimizing a triplet loss over them, which can generate more informative hash codes that can well preserve the semantic relations. Progressively, the output hash codes can serve as the new input to the GAN, guiding the following repeated learning process, towards much better hash codes.

To the best of our knowledge, this is the first work that truly pursues deep hash functions in the generative adversarial networks under the unsupervised setting and achieves the state-of-the-art performance over several popular image datasets.

2 Progressive Generative Hashing

In this section, we will elaborate our Progressive Generative Hashing (PGH) model. Before that we first introduce the basic notations used through this paper.

Supposing we have n training images in the training set $\mathcal{X} = \{\mathbf{x}_i | i = 1, \ldots, n\}$, our goal is to learn a deep hashing network H without any label information. The hashing network H can encode each input image $\mathbf{x}_i$ into a compact binary code $\mathbf{b}_i \in \{0, 1\}^k$ in an end-to-end way, where k is the code length. Intuitively, the generated binary codes $\mathbf{B} = \{\mathbf{b}_i | i = 1, \ldots, n\}$ are expected to preserve the similarity relationship among the images. To achieve this goal, we employ a generative adversarial architecture, consisting of a generative network G and a discriminative network D, to generate complementary images for training H. Figure 1 illustrates the architecture of the proposed PGH method.

2.1 Hash Conditioned GANs

Learning a deep hashing network usually requires sufficient training images that can convey the intrinsic distribution. Without label information, to learn a discriminative H, we have to resort to certain way to synthesize complementary images that can uncover the underlying structure of the data, together with the original training images.

The generative adversarial architecture has been proved to be a promising solution, owing to its surprising capability of generating the realistic images. However, without any guidance to the desired hash codes, such a solution might diverge from the goal due to its unstable training. Since most of existing hashing methods attempted to preserve the neighbor relations, the generated compact hash codes, which can distinguish the similar data with a large probability, can be viewed as a kind of weak semantic information.

Based on the fact, we propose a hash conditioned generative adversarial networks, where both G and D are conditioned on certain binary codes generated by any weak unsupervised hashing method. This is similar to the idea used in conditional GANs [Mirza and Osindero, 2014] in the literature, where their generator and discriminator are usually conditioned on some semantic information like class labels. Our model largely differs from them by simply utilizing the easily obtained weak hash codes without heavy requirements on the strong semantic clues.

Even though such inferior binary codes may not be discriminative enough for image retrieval tasks, they can provide auxiliary information for generative adversarial networks to produce plausible and semantically similar images. Besides, they can bring extra information that can supervise and thus stabilize the hard-to-train generative adversarial network.

In particular, for a training image $\mathbf{x}_i$, the generator network G takes a random noise vector $\mathbf{z}_i$ and an initial weak binary code $\mathbf{b}_i$ as the input and generates a synthetic image

$$\bar{\mathbf{x}}_i = \mathrm{G}(\mathbf{z}_i | \mathbf{b}_i).$$

Such synthetic images should preserve the original similarity of the input hash codes. Namely, if the Hamming distance between two binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ is greater than that between $\mathbf{b}_i$ and $\mathbf{b}_k$, then the synthetic image $\bar{\mathbf{x}}_j$ should be more similar to $\bar{\mathbf{x}}_i$ than $\bar{\mathbf{x}}_k$.

For the discriminator D, the real image $\mathbf{x}_i$ and the generated one $\bar{\mathbf{x}}_i$ will be respectively combined with the corresponding binary codes $\mathbf{b}_i$ as the input. D will evaluate the probability that the input comes from the training data rather than G. Therefore, the whole GANs can be trained in a two-player min-max game. Specifically, given an image sample $\mathbf{x}_i$ and its weak binary code $\mathbf{b}_i$, the objective function of the two-player min-max game would be:

$$\min_{\mathrm{G}} \max_{\mathrm{D}} \mathcal{L}_{\mathrm{D,G}} = \log \mathrm{D}(\mathbf{x}_i | \mathbf{b}_i) + \log(1 - \mathrm{D} \circ \mathrm{G}(\mathbf{z}_i | \mathbf{b}_i)) \quad (1)$$

2.2 Deep Hashing Network

In our paper, we mainly focus on the unsupervised setting without any label information. Therefore, to learn a discriminative H, we only resort to the training data and its inherent structure. Fortunately, the hash conditioned GANs are able to discover the structure and thus provide more explicit information by the generated images. Since it is reasonable to expect that a real image $\mathbf{x}_i$ is more similar to its corresponding generated image $\bar{\mathbf{x}}_i$ than other synthetic images, we can form real-synthetic triplets using the original real images and their generated images.

Specifically, with the help of our hash conditioned GANs we can easily obtain a set of real-synthetic triplets, where

each triplet $(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ consists of a real image $\mathbf{x}_i$, its synthetic one $\bar{\mathbf{x}}_i$, and any other synthetic one $\bar{\mathbf{x}}_j$ $(j \neq i)$. Note that $\bar{\mathbf{x}}_i = \mathrm{G}(\mathbf{z}_i|\mathbf{b}_i)$, which means it is synthesized by generator network G conditioned on the weak binary code of $\mathbf{x}_i$, and thus has much stronger similarity with $\mathbf{x}_i$ than any $\bar{\mathbf{x}}_j$ generated upon the different images and hash codes.

Intuitively, the desired hash codes should be able to preserve the relative similarities among the images in the triplet. In this way, we can define a triplet loss that has also been successfully applied in prior research [Qiu *et al.*, 2017], which is defined over the output binary codes $\mathbf{b}_i$, $\bar{\mathbf{b}}_i$ and $\bar{\mathbf{b}}_j$ corresponding to the training image triplet $(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$

$$\max(0, \gamma k - \|\mathbf{b}_i - \bar{\mathbf{b}}_j\|_{\mathcal{H}} + \|\mathbf{b}_i - \bar{\mathbf{b}}_i\|_{\mathcal{H}}),$$

where $\|\cdot\|_{\mathcal{H}}$ represents Hamming distance and the parameter γ controls the minimum Hamming distance margin between the similar and dissimilar image pairs.

To pursue such discriminative hash codes, we can learn the hashing network H by minimizing the triplet loss. This can drive H to possess strong capability of distinguishing the images with slight changes. Since the hash codes are discrete, we simply adopt the following quantization solution to generate hash codes using H:

$$\mathbf{b}_i = \frac{1}{2}\mathrm{sgn}\left(\mathrm{H}(\mathbf{x}_i) - \frac{1}{2}\right) + \frac{1}{2}. \tag{2}$$

where the output of H belongs to $[0, 1]$.

Since directly minimizing $h(\mathbf{b}_i, \bar{\mathbf{b}}_i, \bar{\mathbf{b}}_j)$ is quite difficult, mainly due to the involvement of the discrete binary codes. To approximately solve the problem, we first can naturally relax the problem by replacing $\mathbf{b}_i$ with $\mathrm{H}(\mathbf{x}_i)$, and simultaneously turn Hamming distance to the l_1 distance. Therefore, the triplet loss can be defined as

$$h(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \max(0, \gamma k - |\mathrm{H}(\mathbf{x}_i) - \mathrm{H}(\bar{\mathbf{x}}_j)| + |\mathrm{H}(\mathbf{x}_i) - \mathrm{H}(\bar{\mathbf{x}}_i)|),$$

To further force the relaxation to be consistent with the discrete hash codes, we need to minimize the quantization loss

$$q(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \|\mathrm{H}(\mathbf{x}_i) - \mathbf{b}_i\|_2^2 + \|\mathrm{H}(\bar{\mathbf{x}}_i) - \bar{\mathbf{b}}_i\|_2^2 + \|\mathrm{H}(\bar{\mathbf{x}}_j) - \bar{\mathbf{b}}_j\|_2^2$$

Based on the two types of loss, we can learn the deep hashing network by solving the following problem

$$\begin{aligned} \min_{\mathrm{H},\mathbf{B},\bar{\mathbf{B}}} \mathcal{L}_{\mathrm{H,G}} &= \sum_{i,j\neq i} h(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) + q(\mathbf{x}_i, \bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) \\ s.t.\ \mathbf{B}, \bar{\mathbf{B}} &\in \{0,1\}^{k\times n} \end{aligned} \tag{3}$$

which can be efficiently solved by alternating the optimization of the subproblem with respect to each variable while fixing the others. Note that besides generating hash codes, the hashing network together with D can also help improve G for better image generation.

2.3 Progressive Architecture

The initial input binary codes might not be able to accurately characterize the inherent relations among the training images. However, with the triplet loss guiding deep hashing network H, it is possible for us to pursue better binary codes by absorbing more information from both the real images and the synthetic ones. We will later show this fact indeed holds in our experimental part.

Under the assumption that the quality of the binary codes can be improved through our hash conditioned GANs and the hashing network, we can naturally introduce a progressive architecture to feed the improved hash codes into the networks again, and thus continuously increase the code quality, establishing a learning cycle until converging. In the progressive architecture, as shown in Figure 1, the binary codes connect the two successive learning cycle, *i.e.*, the output hash ones, learnt upon the input weak ones in the last round, will be returned as the new input to start the next round learning. By repeating such process, we gradually achieve more realistic image generation and a more discriminative hash function.

Specifically, at the t-th loop we input the weak binary codes $\mathbf{B}^{(t-1)}$, $t = 1, \ldots, T$ to G and D, and output new binary ones $\mathbf{B}^{(t)}$ by H. We start the progressive learning by $\mathbf{B}^{(0)}$ generated by any weak hashing method. Then, for the t-th loop, the overall objective function in our progressive architecture, with respect to current networks $\mathrm{G}^{(t)}$, $\mathrm{D}^{(t)}$, $\mathrm{H}^{(t)}$ and the desired binary codes $\mathbf{B}^{(t)}$, should be

$$\begin{aligned} \min_{\mathrm{G}^{(t)},\mathrm{H}^{(t)}} \max_{\mathrm{D}^{(t)},\mathbf{B}^{(t)},\bar{\mathbf{B}}^{(t)}} & \mathcal{L}_{\mathrm{D}^{(t)},\mathrm{G}^{(t)}} + \lambda\mathcal{L}_{\mathrm{H}^{(t)},\mathrm{G}^{(t)}} \\ s.t. \qquad & \mathbf{B}, \bar{\mathbf{B}} \in \{0,1\}^{k\times n} \end{aligned} \tag{4}$$

2.4 Optimization Details

In the above problem the desired networks and the binary codes can be easily optimized one by one iteratively. However, to achieve fast convergence and satisfying performance, we also should pay much attentions to the training order of the different networks. Since the synthetic images play an important role on the hashing network learning, to guarantee their quality, practically we make sure hash conditioned generative adversarial networks (G and D) are adequately trained first. This can be completed by solving the subproblem in Equation (1) without considering the effect of hashing network. After that, we can simultaneously take the three networks into consideration in the next training process by solving the problem in Equation (4) for each progressive stage. Algorithm 1 lists the main steps of our PGH method.

In the above training process, we believe the hashing network could extract certain semantic information, which in turn will help stabilize and improve the performance of the generator. We find that when training generator G in our method, to emphasize the semantic similarity between real images and synthetic ones, replacing the triplet loss with the following loss can further the performance in practice:

$$\mathcal{L}_{\mathrm{H,G}} = \sum_{i=1}^{n} \|\mathrm{H}^{(t-1)}(\mathbf{x}_i) - \mathrm{H}^{(t-1)} \circ \mathrm{G}^{(t)}(\mathbf{z}_j|\mathbf{b}_j^{(t-1)}))\|_2^2$$

3 Experiments

In this section, we will evaluate the proposed progressive generative hashing (PGH) method in the task of image retrieval. We adopt two widely used large image datasets, *i.e.*,

Algorithm 1 Progressive Generative Hashing (PGH)

Input: training set $\mathcal{X} = \{\mathbf{x}_i | i = 1, \ldots, n\}$, weak codes set $\mathcal{B} = \{\mathbf{b}_i | i = 1, \ldots, n\}$
Output: non-linear parameters set $\mathcal{W}_{\mathrm{D}}$, $\mathcal{W}_{\mathrm{H}}$ and $\mathcal{W}_{\mathrm{G}}$

```
for number of progressive loops do
    for number of training epochs do
        for k_1 steps do
            sample minibatch of m noise {z_1, z_2, ..., z_m}
            sample minibatch of m examples {x_1, x_2, ..., x_m}
            optimize W_D to max_D L_{D,G}
        end for
        for k_2 steps do
            construct minibatch of m tuples {(x_i, x̄_i, x̄_j)}
            optimize W_H to min_H L_{H,G}
        end for
        sample minibatch of m noise {z_1, z_2, ..., z_m}
        sample minibatch of m examples {x_1, x_2, ..., x_m}
        construct minibatch of m tuples {(x_i, x̄_i, x̄_j)}
        optimize W_G to min_G L_{D,G} + λ L_{H,G}
    end for
    update B with outputs of H
end for
Return W_D, W_H and W_G
```

Standard gradient-based optimization methods can be used to learn $\mathcal{W}_{\mathrm{D}}$, $\mathcal{W}_{\mathrm{H}}$ and $\mathcal{W}_{\mathrm{G}}$

METHODS	mAP (%)		
	$b = 16$	$b = 32$	$b = 64$
LSH+PIXEL	20.52	26.02	32.14
ITQ+PIXEL	39.57	42.98	44.88
SH+PIXEL	26.60	35.92	25.01
SPH+PIXEL	26.49	31.03	35.59
LSH+GIST	22.75	29.08	33.95
ITQ+GIST	38.64	43.12	45.66
SH+GIST	29.76	30.55	28.50
SPH+GIST	31.15	35.57	39.27
DH+GIST	**43.14**	44.94	46.74
DEEPBIT	28.18	32.02	44.53
BGAN+GIST	40.26	40.78	51.61
PGH+GIST	39.20	**66.95**	**67.95**

Table 1: Performance of different methods on MNIST.

METHODS	mAP (%)		
	$b = 16$	$b = 32$	$b = 64$
LSH+GIST	12.78	13.97	15.07
ITQ+GIST	16.36	17.00	17.58
SH+GIST	13.06	12.92	13.07
SPH+GIST	14.63	15.14	15.86
LSH+VGG	15.38	17.17	20.73
ITQ+VGG	25.51	26.57	28.23
SH+VGG	17.29	16.44	16.56
SPH+VGG	18.97	21.26	22.97
DH+GIST	16.17	16.62	16.96
DEEPBIT	19.43	24.86	27.73
BGAN+GIST	19.15	19.42	21.60
PGH+GIST	**33.40**	**33.98**	**32.40**

Table 2: Performance of different methods on CIFAR-10.

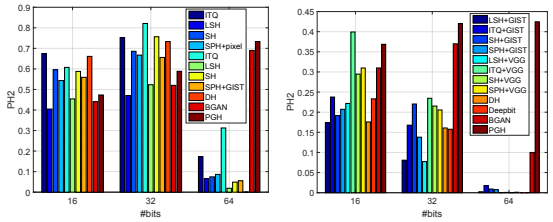
MNIST [LeCun *et al.*, 1998] and CIFAR-10 [Krizhevsky, 2009]. MNIST is a famous handwritten digits dataset containing a training set of 60,000 examples, and a test set of 10,000 examples. It has been widely used for computer vision tasks to test the generalization of different algorithms. CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images, respectively.

Since we mainly focus on the unsupervised learning, we compare our method only with the state-of-the-art unsupervised hashing methods, including (1) the shallow models: Local Sensitive Hashing (LSH) [Datar *et al.*, 2004], Iterative Quantization (ITQ) [Gong and Lazebnik, 2011], Spectral Hashing (SH) [Salakhutdinov and Hinton, 2009] and Spherical Hashing (SPH) [Heo *et al.*, 2015]; (2) the deep models: Deep Hashing (DH) [Liong *et al.*, 2015] and Deepbit [Lin *et al.*, 2016]. Both DH and Deepbit are the most recent unsupervised deep hashing solutions. Note that DH doesn't take advantages of the deep convolutional neural network, which simply generates hash codes based on the handcrafted features. We also compare our method with BGAN [Song, 2017] that also utilizes GANs for hashing, but heavily relies on the similarity matrix derived from certain type of features. To make the comparison fair under the settings without any supervised information, for DH, BGAN and PGH we adopt the same type of low-level features, instead of deep features like ResNet trained in a supervised way.

3.1 Implementation and Evaluation Protocols

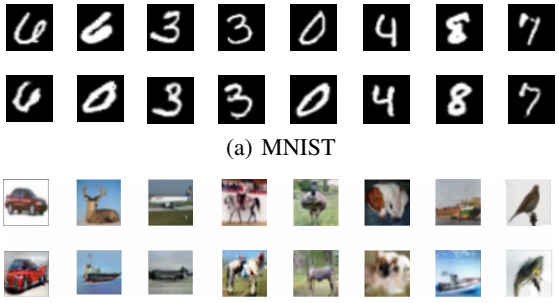
We implement our PGH model using Pytorch, whose architecture is shown in Figure 1. For the hash conditioned GANs, we adopt a combination of DCGAN and CGAN, where the former improves the quality of synthetic images while the latter provides ability of generating similar and dissimilar samples to specific image. We comply with the practical advices in [Radford *et al.*, 2015] and [Mirza and Osindero, 2014], and make certain necessary modification to fit our problem. As to the training, in all experiments we employ the Adam optimizer with a mini-batch size of 64. Besides, we fix the learning rate and the momentum to 0.0002 and 0.9, respectively. For the hashing network, we simply adopt the popular VGGNet [Simonyan and Zisserman, 2014], where to generate hash codes we replace the softmax function after the last fully-connected layer with the quantization function in Equation (2). Also we choose the sigmoid as our active function, and the stochastic gradient descent algorithm for the optimization. As to the utilization, on CIFAR-10 we borrow the pre-trained weights from the 16 layers VGGNet, which is trained on the similar large scale natural image dataset ImageNet. We set the start learning rate to 0.0003, and decrease it by 10% every 10 epochs. The mini-batch size is 32, and each image is normalized and rescaled to 224×224 as the network input. Different from CIFAR-10, on MNIST we train the hashing network from scratch. The start learning rate is 0.002, decreased by 10% every 20 epochs. Here, we use mini-batch size of 64, and all images are normalized to $[-1, 1]$.

To comprehensively evaluate the hashing performance of different methods, we adopt two common search schemes including the Hamming distance ranking and the hash table lookup. The Hamming distance ranking places those points that have small Hamming distances to the query on the top of the result list. We report the ranking performance on this list in terms of mean average precision (mAP). Hash table lookup works like the popular inverted indexing, where the points falling within a small Hamming radius r (usually $r \leq 2$ for efficiency) from the query code are retrieved as the final results. We choose the widely used precision within Hamming



(a) MNIST (b) CIFAR-10

Figure 2: PH2 performance on MNIST and CIFAR-10.



(a) MNIST

(b) CIFAR-10

Figure 3: Visualization of real and synthetic images

radius 2 (PH2) to evaluate the search performance.

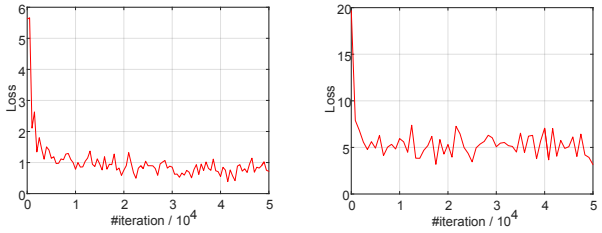
3.2 Results and Discussions

We will first investigate the performance of our PGH model on image search, compared to the state-of-the-art unsupervised hashing methods. Then we will analyze our model from the aspects of the convergence and progressive improvement.

Image Search Performance

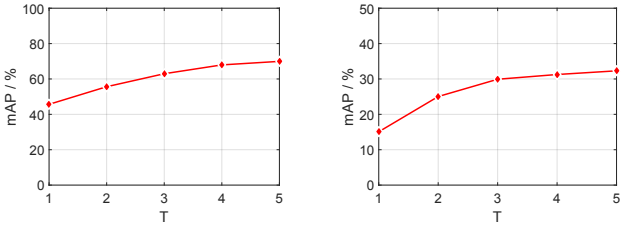
In the image search experiments, we compare our method with the shallow hashing methods using the hand-crafted features and deep features. On CIFAR-10 we adopt both 512-D GIST features and 4096-D deep features extracted by VGGNet. On MNIST, since deep features are not applicable to handwriting digits in the unsupervised setting, we simply choose the pixel-level features and the GIST features.

Table 1 reports the mAP performance with respect to different number of hash bits. From the table, we can observe that the shallow hashing methods based on pixel features and GIST features achieve comparable performance at a similar level in most cases, compared to the deep methods. Besides, it should be noted that the performance of ITQ is quite close to the deep hashing methods including DH, Deepbit and BGAN. Since DH also takes GIST features as its input, we believe that the low-level hand-crafted features lack the capability of describing the underlying structure of the digital images in MNIST. Besides, we find that Deepbit also fails to obtain significant performance gains over the shallow hashing methods. BGAN can achieve satisfying performance in most cases, guided by a similarity matrix based on GIST features. Our PGH also relies on GIST features, and by inputting the hash codes generated by ITQ on GIST features, it can consistently perform best in most cases. The result proves that PGH owns encouraging ability to pursue more discriminative codes than the input weak ones in the progressive learning.



(a) MNIST (b) CIFAR-10

Figure 4: Training loss of PGH on MNIST and CIFAR-10.



(a) MNIST (b) CIFAR-10

Figure 5: The progressive learning curves of PGH.

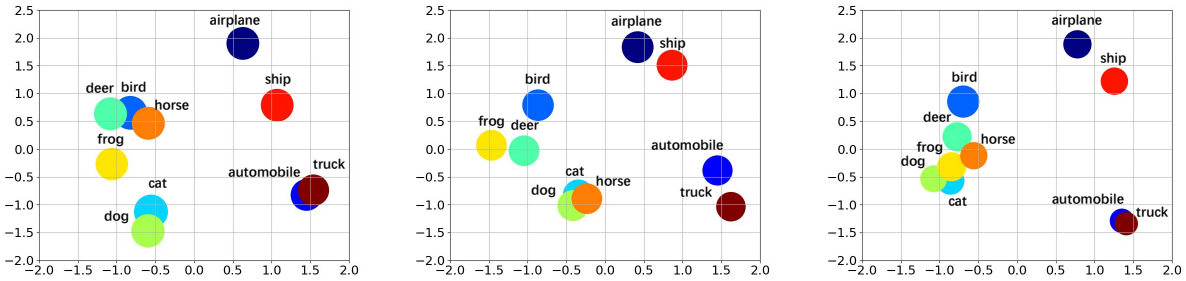
We further study the performance of different methods on CIFAR-10. In this experiments, we first compare the deep hashing methods with the shallow ones with hand-crafted features and deep ones. As the results show, the shallow models with deep features can beat the deep ones, and significantly outperform them with hand-crafted features. For example, ITQ with VGG features can perform better than Deepbit and BGAN in all cases. This phenomenon indicates that the conventional unsupervised hashing methods can also work well if equipped with a good feature representation. On the other hand, it also means that existing deep hashing methods have not fully exploited the hashing nature. From this point of view, we believe that our PGH serves as a novel alternative to discover a discriminative coding solution that can capture the semantic differences between the images, mainly relying on the image triplets generated by the hash conditioned GANs. Therefore, on CIFAR-10 PGH obtains the best performance, with significantly performance gains over the second best.

Model Analysis

Besides the Hamming distance ranking performance, Figure 2 also depicts the PH2 performance on MNIST and CIFAR-10. Here, PGH gives a relatively lower performance when using short binary codes. But we can see that it can largely increase its precision when using more hash bits while others drop quickly, which means that PGH can preserve the semantic relations using the progressively enhanced hash codes, mainly owing to the realistic synthetic images. We will further illustrate this point in the following experiments.

In Figure 3 we show several synthetic images generated by our hash conditioned GANs on MNIST and CIFAR-10, where each subfigure presents the real images and the corresponding synthetic ones respectively at the top and bottom rows. We can easily observe that our framework can provide images with the same semantic but slightly varied, which certainly helps enrich the neighbor relations for the code learning and improve the robustness of the hashing network.

Convergence. Traditional GANs usually suffer from the un-



(a) ITQ (b) Deepbit (c) PGH

Figure 6: The distribution of 10 classes based on hash codes generated by different methods on CIFAR-10.

METHODS	mAP (%)				
	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
LSH+GIST	15.07	25.03	29.92	31.26	32.18
ITQ+GIST	17.58	27.04	30.49	31.81	32.40
ITQ+VGG	28.23	32.93	33.30	-	-

Table 3: The progressive performance using different input hash codes on CIFAR-10.

stable convergence in practice. Here, our hash conditioned GANs enjoy stable and fast convergence at the training stage. This is mainly because the weak hash codes can help avoid the undesired mode collapse and thus lead the networks to approximate the true semantic distribution of the images. Figure 4 shows the convergence curves on MNIST and CIFAR-10, where we mainly depict the variation tendency after the three networks are trained at the same time.

Progressive Improvement. Besides the fast convergence of hash conditioned GANs, we also want to investigate the property of our whole progressive learning framework. To demonstrate PGH can progressively learn more discriminative hash codes based on the weak hash codes, in Figure 5 (a) and (b) we respectively show how the performance varies on MNIST and CIFAR-10 using 64 bits, with respect to the number T of the progressive iterations. It is obvious that on both datasets the mAP performance increases significantly when we iteratively input the learnt hash codes of last iteration into the current hash conditioned GANs. Therefore, we can conclude that our PGH can learn desired hash codes quickly based on the weak ones and improve the performance gradually. In practice, usually the progressive process can stop in 4-5 iterations when it reaches the convergency.

Intuitively, different input hash codes provide different condition information for image generation using hash conditioned GANs. For example, ITQ with the VGG deep features will transmit more rich semantics into the model than LSH with low level GIST features. To check the effect of the input codes, we consider three different but typical initial input hash codes to start PGH learning. In Table 3, we report the progressive performance of different input hash codes on CIFAR-10. First, we observe that in all cases PGH consistently improves the performance iteration by iteration. More importantly, even though the performance of LSH with GIST features is inferior to that of ITQ with VGG features, we can notably see that all the three types of codes almost converge to the same level of performance in a very few iterations. This means that our model doesn't highly depend on the initial input, but can perform robustly and steadily in practice.

Semantic Discovery. We have pointed out and validated in the prior experiments that our PGH can synthesize semantically realistic images, which together with the original training images are able to discover the underlying semantic similarities among the data. To further verify this conclusion, In Figure 6 (a)-(c) we respectively show the distributions of the 10 classes on CIFAR-10 in 2-D space by PCA, according to the 64-bit hash codes generated by ITQ, Deepbit and our PGH. In these figures, each circle in different color corresponds to a unique semantic class, and its size indicates the variance of hash codes belonging to the class. Compared with ITQ and Deepbit, the best shallow and deep hashing methods as we know, our PGH has the largest inter-similarities among images in the same class, *i.e.*, the average code variance of ten classes is 0.1578, which is much smaller than 0.2141 of ITQ and 0.2010 of Deepbit. Moreover, it can discover the latent semantic similarities among different classes as we expect. Compared with ITQ and Deepbit, PGH owns much stronger capability to distinguish the dissimilar images such as 'bird' images from 'deer' and 'horse' with relatively large distances in Figure 6(c), and exploits the semantically similar ones, such as 'automobile' and 'truck', 'deer' and 'horse', etc., with close hash codes. This further proves that our progressive generative strategy can significantly boost the hashing performance even without supervision information.

4 Conclusion

In this paper we presented a novel unsupervised progressive generative hashing (PGH) architecture, which exploits the power of hash conditioned GANs and the progressive learning. The hash conditioned GANs take weak binary codes to guide image generation, and help train a discriminative hash function based on the triplet loss defined over both real and synthetic images. Through progressively feeding the learned binary codes into the network, we can obtain better deep hash function gradually. Experiments and analysis have demonstrated that PGH can discover the semantic relations, and thus achieve the state-of-the-art performance.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61690202) and MSRA Collaborative Research Grant.

References

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262, 2004.

[Gong and Lazebnik, 2011] Yunchao Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE CVPR*, pages 817–824, 2011.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pougetabadie, Mehdi Mirza, Bing Xu, David Wardefarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 3:2672–2680, 2014.

[He *et al.*, 2012] Junfeng He, Jinyuan Feng, Xianglong Liu, Tao Cheng, Tai-Hsu Lin, Hyunjin Chung, and Shih-Fu Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE CVPR*, pages 3005–3012, 2012.

[He *et al.*, 2013] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE CVPR*, pages 2938–2945, 2013.

[Heo *et al.*, 2015] Jae Pil Heo, Junfeng He, Junfeng He, Shih Fu Chang, and Sung Eui Yoon. Spherical hashing: Binary code embedding with hyperspheres. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 37(11):2304–2316, 2015.

[Indyk, 1998] Piotr Indyk. Approximate nearest neighbors: towards removing the curse of dimensionality. *Theory of Computing*, 604-613(11):604–613, 1998.

[Kong and Li, 2012] Weihao Kong and Wu-Jun Li. Isotropic hashing. In *NIPS*, pages 1–8, 2012.

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

[Kulis and Darrell, 2009] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1–8, 2009.

[LeCun *et al.*, 1998] Y. L. LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. proc ieee. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lin *et al.*, 2016] Kevin Lin, Jiwen Lu, Chu Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *IEEE CVPR*, pages 1183–1192, 2016.

[Liong *et al.*, 2015] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *IEEE CVPR*, pages 2475–2483, 2015.

[Liu *et al.*, 2011] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[Liu *et al.*, 2013] Xianglong Liu, Junfeng He, Bo Lang, and Shih-Fu Chang. Hash bit selection: a unified solution for selection problems in hashing. In *IEEE CVPR*, 2013.

[Liu *et al.*, 2014a] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[Liu *et al.*, 2014b] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *IEEE CVPR*, pages 2147–2154, 2014.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *Computer Science*, pages 2672–2680, 2014.

[Mu *et al.*, 2014] Yadong Mu, Gang Hua, Wei Fan, and Shih-Fu Chang. Hash-svm: Scalable kernel machines for large-scale visual classification. In *IEEE CVPR*, pages 979–986, 2014.

[Qiu *et al.*, 2017] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. Deep semantic hashing with generative adversarial networks. In *ACM SIGIR*, pages 225–234, 2017.

[Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Computer Science*, 2015.

[Raginsky and Lazebnik, 2009] Maxim Raginsky and Svetlana Lazebnik. Local-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1–8, 2009.

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *IEEE CVPR*, pages 37–45, 2015.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.

[Song, 2017] Jingkuan Song. Binary generative adversarial networks for image retrieval. In *AAAI*, 2017.

[Wang *et al.*, 2014] Qi Wang, Guokang Zhu, and Yuan Yuan. Statistical quantization for similarity search. *Computer Vision and Image Understanding*, 124:22–30, 2014.

[Wang *et al.*, 2015] Qifan Wang, Luo Si, and Bin Shen. Learning to hash on structured data. In *AAAI*, pages 3066–3072, 2015.

[Wang *et al.*, 2017] Qi Wang, Jia Wan, and Yuan Yuan. Locality constraint distance metric learning for traffic congestion detection. *Pattern Recognition*, 75, 2017.

[Wang *et al.*, 2018] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[Weiss *et al.*, 2008] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1–8, 2008.

[Yu *et al.*, 2014] X. Yu, S. Kumar, Yunchao Gong, and S.F. Chang. Circulant binary embedding. In *ICML*, pages 1–8, 2014.