Deep Semantic Hashing with Multi-Adversarial Training

Bingning Wang†
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
bingning.wang@nlpr.ia.ac.cn

Kang Liu†
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
kliu@nlpr.ia.ac.cn

Jun Zhao
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
jzhao@nlpr.ia.ac.cn

ABSTRACT

With the amount of data has been rapidly growing over the recent decades, binary hashing has become an attractive approach for fast search over large databases, in which the high-dimensional data such as image, video or text is mapped into a low-dimensional binary code. Searching in this hamming space is extremely efficient which is independent of the data size. A lot of methods have been proposed to learn this binary mapping. However, to make the binary codes conserve the input information, previous works mostly resort to mean squared error, which is prone to lose a lot of input information [11]. On the other hand, most of the previous works adopt the norm constraint or approximation on the hidden representation to make it as close as possible to binary, but the norm constraint is too strict that harms the expressiveness and flexibility of the code.

In this paper, to generate desirable binary codes, we introduce two adversarial training procedures to the hashing process. We replace the $L_2$ reconstruction error with an adversarial training process to make the codes reserve its input information, and we apply another adversarial learning discriminator on the hidden codes to make it proximate to binary. With the adversarial training process, the generated codes are getting close to binary while also conserves the input information. We conduct comprehensive experiments on both supervised and unsupervised hashing applications and achieves a new state of the arts result on many image hashing benchmarks.

CCS CONCEPTS

- Computing methodologies → Neural networks; 
- Security and privacy → Hash functions and message authentication codes;

KEYWORDS

Semantic Hashing; Generative adversarial network; Auto-encoder; Unsupervised Learning

†Bingning and Kang have equal contributions to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:


1 INTRODUCTION

Similarity search has many important applications such as document clustering, multimedia retrieval, and collaborative filtering [46]. In practical scenarios, especially when the data is very large, scalable retrieval is needed to accelerate the searching process. However, searching is very challenging since the feature space of data, such as images, usually span very high dimension. Operation such as calculating cosine distance in this high dimension is extremely computationally expensive, which requires a more efficient scheme for searching. To this end, semantic hashing, one of the most effective ways to accelerate the searching process through approximate matching based on binary code representation, has been widely studied and became the mainstream technique [14].

The basic idea of hashing is to map high dimensional data into low-dimension binary codes through a series of hash functions. Based on such binary representation, similarity could be efficiently calculated. Traditional methods for hashing such as locality-sensitive hashing (LSH) [5, 22], semantic hashing [39], spectral hashing [48] have been proved to be very useful for constructing such hash functions. However, they are either data-independent or requiring specific assumptions about the underlying distributions of the data, which hinder their applications in large-scale scenario [46].

Recently, with the renaissance of the neural networks in artificial intelligence, some deep learning based auto-encoder have been proposed for semantic hashing [3, 10, 32]. They usually adopt a neural network based encoder to embed the input into a low-dimensional continuous hidden representation $h$. And a decoder is used to build a reconstruction $\tilde{x}$ from $h$. As a result, $h$ could be served as a surrogate of the input and used for hashing. Basically, these methods are based on two objectives: (I) the hidden representation $h$ should preserve the semantic information of $x$ as much as possible. (II) $h$ should be as close as possible to binary.

In order to preserve the semantic information of input data, previous methods usually adopt norm constraint to make the reconstruction similar with the input, such as $L_2$ norm (mean squared error) or $L_1$ norm [3, 10]. The objective of reconstruction is $\|x - \tilde{x}\|_2^2$. However, it has been proved that simply employing the $L_1$ or $L_2$ norm is unable to capture the structure information of the input [12, 30], so the reconstruction is similar with the input only in a global view and loses a lot of details [11].
On the other hand, to approximate the hidden representations $(\mathbf{h})$ to binary codes, most previous approaches also adopt norm-constraint. For example, Binary Auto-encoders [3] use the $L_2$ norm to minimize the distance between $\mathbf{h}$ and its sign value (i.e. $||\text{sign}(\mathbf{h}) - \mathbf{h}||_2$). However, the sign function is non-differentiable, it needs to be fine-tuned and unstable when the code size is large [7]. To overcome such problem, some methods directly set a hard threshold to $\mathbf{h}$ to derive the binary codes [4, 47]. This strategy simplifies the optimization greatly but usually yield low-quality solutions [8].

In this paper, to generate better binary codes without any norm constraint, we propose a deep semantic hashing model based on generative adversarial networks (GANs) [17]. The whole architecture is based on an auto-encoder, which served as the generator in GANs. Then we use two discriminators as the adversarial components for the auto-encoder to learn the distance information, which bypasses the traditional norm constraints. We refer our model as generative multi-adversarial networks (GMANs).

Specifically, to make the reconstruction of auto-encoder has more fidelity, we use a discriminator to determine whether the input is fake (generated by the auto-encoder) or real (original input), and the output from such discriminator could be used as the distance information for training the auto-encoder. In this way, rather than employing the inefficient $L_2$ or $L_1$ loss, the auto-encoder is trained to minimize the Jensen-Shannon divergence as the reconstruction objective [17].

Secondly, we apply another discriminator to the hidden representation $\mathbf{h}$ of the auto-encoder to make $\mathbf{h}$ as close as possible to be binary. We randomly sample a binary code from a specified Bernoulli distribution and treat it as the real data for this discriminator. Meanwhile, the generated codes $\mathbf{h}$ are served as the fake data to the discriminator. In this case, the objective of the discriminator is to determine whether the input is binary or not. With such adversarial training, the generated codes from the auto-encoder are expected to get close to the real binary data.

Unfortunately, the GANs are notorious for training [40]. One of the most challenging problems is the mode collapse issue [42, 45]. Therefore, in this paper, we employ the variance of the generated codes as an additional regulator to train the generator, which could effectively alleviate the mode collapse problem. Besides, to make the adversarial training process more smooth and stable, we further propose a self-controlling scheme where the generator and discriminator are trained dynamically so that the generator could get the most substantial gradient from discriminator.

Our proposed model is based on auto-encoder which is purely unsupervised. However, it is straightforward to apply our model to a supervised scenario. We apply GMANs on the widely used image and text hashing datasets. Experimental results show the effectiveness of the proposed approach. We conduct further experiments to demonstrate the specific advantage of each adversarial network in our model. The results show that GMANs could yield better reconstructions compared with the previous method trained with the norm constraint. Moreover, the codes generated by GMANs could preserve the input information and are very close to binary. Although the proposed model is based on auto-encoder which is purely unsupervised, we further prove that it is straightforward to apply our model on supervised scenario. Briefly, the contributions of this paper can be summarized as follows:

- We proposed a novel semantic hashing model based on a generative multi-adversarial network. It uses two adversarial training to replace norm constraint to generate high-quality hashing codes.
- We proposed a variance control criteria and self-controlling training scheme on our GMANs to stabilize the training process.
- We conduct comprehensive experiments on several retrieval applications. The results show that our model achieves better results than state-of-the-art models on both supervised and unsupervised setups.

2 RELATED WORKS

Hashing has been regarded as an essential component in a variety of large-scale information retrieval systems. Locality-Sensitive Hashing (LSH) [22] is one of the most widely used unsupervised hashing methods with asymptotic theoretical properties of hashing guarantees. It constructs the hash functions based on many random linear projections. However, LSH is data-independent and usually requires many hashing bits to prevent the collision. To deal with this problem, several data-dependent hashing methods based on machine learning has been proposed. For example, Spectral Hashing (SpH) [48] explored the data distribution by preserving the similarity among the input data and adding the balanced and uncorrelated constraints into the learned codes. Iterative quantization [16] introduced structure and orthogonal constraints on the parameters, which could yield better results but also increase the computation complexity. Vector quantization and its generalizations [24, 52] constrained the binary representation to be 1-of-K coding, so that the corresponding optimization is simplified. However, with these extra constraints, the models become too restricted to generate proper binary codes and thus lose some important information of the inputs. Many deep learning based models have been proposed to reconstruct the data through an auto-encoder (AE). The hidden representation of the AE could be utilized as the hashing codes. Salakhutdinov and Hinton [39] proposed a deep learning model by using Restricted Boltzmann Machines (RBMs). This idea was then extended to more general auto-encoders which remove the Boltzmann constraints. For example, Liong et al. [33] proposed a model to learn the binary bits with auto-encoder jointly. However, there are too many norm constraints on the models which make the learning process extremely difficult to converge. Yu et al. [49] used circulant constraints and Zhang et al. [51] introduced Kronecker Product structure to achieve the binary constraints. These constraints could alleviate the convergence problem but substantially reduce the model flexibility. Other methods such as [7, 8] proposed some machine learning techniques to learn good hidden codes, but it necessitates the meticulous hyper-parameter tuning. Chaidaroon and Fang [4] approximated a Gaussian prior on the hidden codes through variational inference. However, the Gaussian distribution is not appropriate for hashing applications where the desiderata of the codes are binary. Qiu et al. [38] is related to our model. They also proposed a GANs based hashing model. Their method is supervised where they use GANs to generate the fake image for pairwise learning. By contrast, the GANs in our model is...
introduced to the hashing process directly, which act as regulators, to make the latent code as binary as possible.

**Generative Adversarial Networks** GANs [17] is a recently proposed framework for estimating generative models via an adversarial process. In GANs, two types of models are simultaneously trained: a generator $G$ is trained to estimate the data distribution and generate a fake sample; a discriminator $D$ is trained to discriminate the real sample from the fake one. GANs corresponds to a minimax two-player game where there exists a unique solution that $G$ recovers the data distribution, and $D$ equals to $\frac{1}{2}$ everywhere. Many applications and improvement have been proposed to GANs. LAPGAN [9] generated images in a coarse-to-fine fashion by generating and upsampling in multiple steps; InfoGAN [6], an information-theoretic extension to the GANs that could learn disentangled representations. [50] extended the GANs to generating texts by policy gradient. WGAN [1] and BSGAN [21] were two typical ameliorations to vanilla GANs that had better training stabilities and theoretical advantages. Compared with other generative models such as variational auto-encoder [26], GANs could generate higher quality images without any distribution hypothesis. Thus it removes the inference process which is very hard to resolve in other generative models. The GANs have also shown advantages to replace or enhance the traditional metric-based methods in generative models. For example, Larsen et al. [30] found that the traditional $L_1$ or $L_2$ objective may cause the reconstruction be too blurred to be recognized. They also adopt the GANs on the learning process, but it was achieved based on a variational-auto-encoder. Dosovitskiy and Brox [11] extended their model by applying the GANs to the raw input. However, due to incomplete information, it must resort to $L_1$ or $L_2$ to learn the input-output correspondence. Most related method often adopts the adversarial learned inference [12], where the hidden state combined with the data is fed to the discriminator. However, their main motivation is to learn a better inference process. In this paper, our objective is to learn a better auto-encoder. Larsen et al. [30] also proposed a GANs framework that contains more than one discriminator, but these discriminators are applied to the same output of the generator. In this paper, the two discriminators are applied to different parts of the generator.

### 3 SEMANTIC HASHING WITH MULTI-ADVERSARIAL TRAINING

Our semantic hashing model is based on generative multi-adversarial networks. There is a generator which is built upon an auto-encoder. Two adversarial training processes are applied to the auto-encoder: the first one is an auto-encoder-GAN(AE-GAN) and the second one is a binary-GAN(B-GAN). AE-GAN is proposed to replace the traditional norm based reconstruction loss to make the hidden codes reserve the input information. B-GAN is proposed to make the generated codes approximate to binary.

We denote the raw input as $x$, the generated hidden representation as $h \in \mathbb{R}^d$, we use $h$ as the hashing codes for the input. The conceptual architecture of GMANs is illustrated in Figure 1. In the following sections, we will refer the $h$ as hidden representation or hidden codes interchangeably, next we detail the components of GMANs.

![Figure 1: The architecture of the proposed model. 0 represents the fake data while 1 denotes the real data. The B-GAN tries to recognize whether the input is a binary (upmost) vector or a generated hidden codes. The AE-GAN tries to recognize whether the input is the real input ($x$) or generated ($\tilde{x}$). With the adversarial learning proceed, $\tilde{x}$ is more and more similar with $x$. Meanwhile the hidden representation is more approximate to be binary.](image_url)

#### 3.1 Generator

In GMANs the generator corresponds to the auto-encoder which consists of two separate models, namely the encoder and decoder. The encoder $e$ embed the raw input $x$ into a low-dimensional hidden representation $h$ and the decoder $f$ tries to decode $x$ from $h$:

$$h = e(x) \quad \tilde{x} = f(h)$$ (1)

We utilize the convolutional neural networks (CNNs) as the building block for our encoder. CNNs have shown great advantages in many applications [35, 43]. We convolute $5 \times 5$ filters $W$ on the input, so the hidden representation of each layer could be denoted as:

$$h_{i,j} = \text{ReLU}(W \otimes X_{i-2; i+2, j-2; j+2} + b)$$ (2)

where $b$ is the bias vector. We did not adopt any pooling operation but apply CNN several times to the hidden representation to embed the input into a vector. The encoding process is illustrated below:

The decoder architecture can be viewed as the reversed process of the encoder. However, the CNN in the decoder is not the original convolutional operations but the transposed convolution\(^1\) that maps the low-dimensional hidden representation to high-dimensional simulated data $\tilde{x}$.

\(^1\)also known as stride-convolution or de-convolution.
3.2 Auto-Encoder GAN

Traditional hashing methods based on auto-encoder often resort to the Euclidean distance between the input and output, which is more commonly referred as mean squared error, as the training objective:

\[ L_{rec} = d(x, \hat{x}) = \|x - \hat{x}\|_2 \] (3)

We call this objective the norm loss. In AE-GAN, we replace this norm loss with an adversarial training loss. A discriminator in AE-GAN tries to discriminate the original data \( x \) from the fake data \( \hat{x} \) generated by the generator. So we can regard the discriminator as a metric function to provide distance\(^2\) information.

In AE-GAN, the generator is the auto-encoder itself that tries to reconstruct the input. The discriminator in AE-GAN (denoted as \( D_{AE} \)) has two inputs: the first one is the hidden representation \( h \) of the generator, and the second one is the real or fake input \( x \) or \( \hat{x} \). The architecture of the \( D_{AE} \) is illustrated in Figure 2. During training, the objective of the \( D_{AE} \) is to minimize the probability of \( \hat{x} \) while increase the probability of \( x \), which can be denoted as:

\[
L_D^{AE} = -\mathbb{E}_{x \sim p(x), h \sim e(x)}[\log D_{AE}(x, h)] - \mathbb{E}_{x \sim p(x), h \sim e(x), \hat{x} \sim f(h)}[\log(1 - D_{AE}(\hat{x}, h))] \] (4)

\( p(x) \) is the data empirical distribution. In this manner, we could use the output of \( D_{AE} \) as the objective for the generator to optimize. The objective of the generator is:

\[
L_G^{AE} = -\mathbb{E}_{x \sim p(x), h \sim e(x)}[\log D_{AE}(x, h)] \] (5)

This objective is minimized only when the reconstructed \( \hat{x} \) match the original input \( x \). However, in practice, the discriminator is a two-class classifier, so it prone to get too far ahead. Because the discriminator’s output is sigmoidal that the gradient of the value function with respect to the discriminator’s output vanishes to zeros, the generator may have a hard time minimizing the value function above. As a workaround, we train the generator of AE-GAN similar with [12] that maximize:

\[
L_G^{AE} = -\mathbb{E}_{x \sim p(x), h \sim e(x)}[\log(1 - D_{AE}(x, h))] - \mathbb{E}_{x \sim p(x), h \sim e(x), \hat{x} \sim f(h)}[\log D_{AE}(\hat{x}, h)] \] (6)

With the training proceed, the reconstruction \( \hat{x} \) could be more and more similar with the input \( x \) at the guidance of discriminator \( D_{AE} \).

3.3 Binary GAN

While the AE-GAN focuses on minimizing the reconstruction error, the objective of the B-GAN is to approximate the hidden representation into binary codes that could be utilized for hashing. Similar to the AE-GAN, the generator in the B-GANs corresponds to the auto-encoder itself.\(^3\) Denote the discriminator in B-GAN as \( D_B \). We use a multi-layer perceptron (MLP) as the building block for \( D_B \).

\(^3\)More mathematically. It could be regarded as the divergence between the distribution supported by data manifold and the density of the generator. The original GANs[17] cast it as Jensen-Shannon divergence while other works [37] extended this to more general \( f \)-divergences.

\(^3\)Although we can use the output of auto-encoder as the input for the generator to generate the binary code, it has a trivial difference with the proposed strategy. The generator has already involved in the auto-encoder, so we use the auto-encoder as the generator.

PAR\G

\[ DB: \text{In this paper, we adopt scaled exponential linear units (SELU)} \]

\[ g(x) = y \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases} \] (7)

\[ DB: \text{Where the } \gamma \text{ and } \alpha \text{ are two hyper-parameters of SELU. The SELU shows the advantage in convergent speed of the MLP compared with other activation functions such as Tanh or Relu.} \]

\[ DB: \text{For B-GAN, the preferred input is the binary codes, so the real input to } DB \text{ is a binary vector drawn from a Bernoulli distribution:} \]

\[ b \sim B(b, p) \] (8)

\[ DB: \text{where } b \text{ is the number of the bits to be sampled and } p \text{ is the probability that corresponding bit being 1. Previous works on hashing usually assume the balanced bit, so it is trivial to set } p \text{ to 0.5 which means half of the bits to be fired, but we can set } p \text{ to any value between 0 and 1 for different applications. } DB \text{ acts as a critic that output higher score when the input is closer to binary. As the adversarial opponents, the generator has to generate more binary codes to achieve a higher score. Similar to AE-GAN, the objective for B-GAN can be denoted as:} \]

\[
L_D^B = -\mathbb{E}_{b \sim B(b, p)}[\log DB(b)] - \mathbb{E}_{x \sim p(x), h \sim e(x)}[\log(1 - DB(h))] \] (9)

\[
L_G^B = -\mathbb{E}_{x \sim p(x), h \sim e(x)}[\log DB(h)] \]
Thus $L_G^B$ is minimized only when $h$ is binary. However, using Equation 9 to train the generator may be problematic. The real input is binary vector, so the difference between the real and fake input is trivial to be discriminated. In other words, it is unlikely that the model manifold and the true Bernoulli distribution’s support have a non-negligible intersection. To alleviate this problem, we add a noise term to the model distribution. In this case, the fake examples fed to $D_B$ could be formulated as:

$$h' = e(x) + \epsilon, \quad \epsilon \sim N(0, \sigma^2 I)$$ (10)

Where $\sigma$ is the standard deviation of the Gaussian noise which was set to 0.1 in the experiment. The main difference is that we will move our generated codes towards the noised data manifold, which can be thought of as transfer a small neighbourhood of samples towards it. This will protect the discriminator against measure zero adversarial examples, and thus push the generator towards a better manifold. More importantly, the gradient of the generator would correspond with the original one (i.e. two times derivative of the Jensen-Shannon divergence)[1]. The architecture of B-GAN is illustrated in Figure 3.

### 3.4 Mode Collapse Elimination

In experiments, we observe that the B-GAN is prone to mode collapse problem, in which the generated codes collapse to few points that are very similar to binary. Thus the binary score of the generated codes $D_B(h)$ is very high, but the variance of different codes is very small. In another word, no matter what the input is, their corresponding codes generated by the auto-encoder is very close to each other. In theory, this problem is attributed to the fact that the minor mode of the data seldom (approaches to zero probability) getting reward from the discriminator. To eliminate this problem, we add another criterion to the auto-encoder to control the variance of the generated codes:

$$L_V = \mathbb{E}_{B,H}[\text{Var}(B) - \text{Var}(H)]^2$$ (11)

where $B$ and $H$ are batches of real binary codes and generated representations respectively. The variance of the code is:

$$\text{Var}(H) = \text{trace}(H^TH)$$ (12)

As we sampled the real binary vector randomly and independently, we can trivially set $\text{Var}(B)$ to $mb$ where $m$ is the batch size. In this way, the generated codes would be as variable as the sampled binary codes, and could somewhat bypass the mode collapse problem. Adding the variance loss to the original loss we get the final objective of the generator (auto-encoder):

$$L_G = \lambda_1 \cdot L_{AE}^G + \lambda_2 \cdot L_{G}^B + \lambda_3 \cdot L_V$$ (13)

where $\lambda_{1,2,3}$ are three hyper parameters to determine the weight of each objective in training.

### 3.5 Self-Controlled Training procedure

Previous works of GANs carry the adversarial training process under a situation of nearly-perfect discriminator, so they update the discriminator more often than the generator which is claimed to push the generator to the optimal region. However, we find that this mechanism would lead to unstable training process. If the discriminator gets to its optimal situation that it could correctly discriminate the real from the fake data, then the output from the discriminator lies in the saturated region of the sigmoid, so the gradient of from the discriminator to the generator has a large variance. Such high variance may seriously infect the training process and make the generator hardly, if not impossible, to be optimized.

In this paper, to make the generator get gradient from the discriminator consistently, we proposed a training strategy where the generator is updated at the best guidance of the discriminator, i.e. the generator could get the largest gradient from the discriminator. Remember that the gradient of the sigmoid function is: $f'(x) = f(x) \cdot (1 - f(x))$, so it peaks on the value of 0.5. When the output value of discriminator is larger or smaller than 0.5, then it’s gradient will decrease. The gradient of the discriminator could be illustrated in Figure 4.

To keep the discriminator at the best situation that could provide good gradient for the generator, we just utilize the score of the fake example $D(h)$ or $D_{AE}(\hat{x}, h)$ to determine whether we should train the generator or discriminator. When the fake data score is larger than 0.5, we train the discriminator. Otherwise, the generator is trained. From this self-controlled process, the generator and the discriminator could be improved synchronously. The overall training process of GMANs is illustrated in Algorithm 1.

### 4 EXPERIMENTS

In this section, we show the overall results of our GMANs on several images and textual retrieval tasks. In the next section, we will make a detailed analysis of the advantages of the proposed model.

#### 4.1 GMANs vs. State-of-the-arts

##### 4.1.1 Datasets

We apply the GMANs to three image datasets and two text datasets that are widely used as the unsupervised benchmarks for semantic hashing [46]. The image datasets include:

- **CIFAR10** [28]: a dataset of 32x32 color images, each images contains one of ten objects. The training set contains 50k images, and the query (testing) set contains 10k images. We use the training set as the retrieval set.
Table 1: Precision at hamming distance less than 2 on several image datasets.

<table>
<thead>
<tr>
<th></th>
<th>CIFAR10</th>
<th>MNIST</th>
<th>SIFT1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>SH [48]</td>
<td>0.39</td>
<td>4.23</td>
<td>14.37</td>
</tr>
<tr>
<td>ITQ [16]</td>
<td>0.51</td>
<td>5.21</td>
<td>17.77</td>
</tr>
<tr>
<td>SPSH [20]</td>
<td>0.43</td>
<td>3.45</td>
<td>13.47</td>
</tr>
<tr>
<td>KMH [18]</td>
<td>0.53</td>
<td>5.99</td>
<td>19.55</td>
</tr>
<tr>
<td>BA [3]</td>
<td>0.55</td>
<td>5.65</td>
<td>20.23</td>
</tr>
<tr>
<td>SPH [20]</td>
<td>0.43</td>
<td>3.45</td>
<td>13.47</td>
</tr>
<tr>
<td>KMH [18]</td>
<td>0.53</td>
<td>5.99</td>
<td>19.55</td>
</tr>
<tr>
<td>BA [3]</td>
<td>0.55</td>
<td>5.65</td>
<td>20.23</td>
</tr>
<tr>
<td>BDNN [10]</td>
<td>0.55</td>
<td>5.79</td>
<td>22.14</td>
</tr>
<tr>
<td>VDSH [4]</td>
<td>0.61</td>
<td>5.99</td>
<td>23.41</td>
</tr>
<tr>
<td>w/o AE-GAN</td>
<td>0.59</td>
<td>5.87</td>
<td>22.18</td>
</tr>
<tr>
<td>w/o B-GAN</td>
<td>0.56</td>
<td>5.89</td>
<td>23.38</td>
</tr>
<tr>
<td>GMANs</td>
<td>0.62</td>
<td>6.06</td>
<td>25.25</td>
</tr>
</tbody>
</table>

Table 2: Precision at hamming distance less than 2 on several text datasets. For 20Newsgroups as the data size is comparatively small, we do not experiment with code length 64.

4.1.2 Experimental Settings. For CIFAR10 and MNIST, as the input is raw images, we use the CNN with stride size 2 as the building block for auto-encoder. For SIFT1M, the input is the hand-crafted SIFT [34] features, so we use the MLP for the auto-encoder, which is similar to the setting in the compared baselines. For CNN architecture, we initialize all the weight parameters by MSRA [19]: an extension to Xavier and suited for ResNet setting. Batch normalization [23] is added to the encoder and decoder CNN layer to accelerate the training process. For MLP, we initialize their weights by Xavier [15]. We train our model based on Adam [25] algorithms. Batch size is set to 16 for CIFAR10 and MNIST, 32 for SIFT1M. For MLP, the layer size $L$ is set to 3 for both encoder and decoder. $\lambda_3$ is set to 0.2 and 0.5 respectively. For all setup, we set the code length (i.e. b) to be 8, 16, 24, 32 and 48, respectively.

For the text data, as the input is the tf-idf features, we build the GMANs based on MLP, the hidden layer size is halved every layer from 128 to 32. As the input size is relatively large (vocab size was set to 2000), we gradually increasing $\lambda_1$, $\lambda_2$ from 0 to 0.1, which

---

means that we first learn a good auto-encoder and then impose the binary constraint on it.

4.1.3 Evaluation Metrics. For image hashing evaluation, we follow the standard setting in unsupervised hashing that used Euclidean space nearest neighbors of each query as the ground truths [3, 10, 46]. For CIFAR10 and MNIST datasets, we use 50 Euclidean nearest neighbors as the ground truths. For SIFT1M we use 5000 Euclidean nearest neighbors as the ground truths. For the document data, the ground truths documents are those that share the same label with the query document.

We adopt three widely used metrics for evaluation: 1) mean Average Precision of top N retrieved item (mAP@N); 2) precision of Hamming radius 2 (precision@2-bit). It measures precision on retrieved images having Hamming distance to query less than 2 (if no images satisfy we set it to zero); 3) Precision of top N (P@N) retrieved items, which measures whether the system could retrieve the ground-truth items in the top-N retrieved items. We apply mAP@N to image and P@N to text, precision@2-bit is evaluated on both text and image. All experiments have been gone through the significant test, i.e., one-tailed paired t-test with a default 95% significance level is used here.

4.1.4 Baselines. We compare GMANs with seven competitive baselines which have been extensively used for unsupervised semantic hashing, including: Spectral Hashing (SH) [48], Iterative Quantization (ITQ) [16], Spherical Hashing (SPH) [20], K-means hashing (KMH) [18], Binary Auto-encoder (BA) [3], Binary Deep Neural Network (BDNN) [10], Variational deep semantic hashing (VDSH) [4]. In addition, to make specific comparison, we also conduct several ablation studies where we: (1) minimizing the reconstruction error by $L_2$ instead of the AE-GAN (w/o AE-GAN) (2) Optimizing the hidden codes to their corresponding binary vector by $L_2$ norm constraints (w/o B-GAN).

The result is shown in Table 1 and Figure 5 and 6.

4.2 Main Results

It could be seen from table 1, 2 and figure 5, 6 that our proposed GMANs consistently outperform previous methods. Especially, from figure 5 we could observe that the advantage is more significant when the code length increase. This is attributed to the fact that when the code length (i.e. the hidden representation size) is large, it becomes more and more difficult for traditional norm constraint to optimize the hidden codes [46]. However, in our GMANs, the distance information is provided by the discriminator which is less vulnerable to the increasing code length.

The ablation study in table 1 and 2 shows that our proposed adversarial training strategy outperforms the traditional models based on norm constraint. For example, in CIFAR10, the performance of the model without AE-GAN drops a lot. The input in CIFAR10 is 32*32 colored image which lies in high space, so the reconstruction by $L_2$ norm in Equation 3 may put no attention on the specific area of the input. On the contrary, our model utilizes deep neural networks to measure this distance, which would capture more important information of the input image. In addition, the model without B-GAN behaves poorly. We find in the experiment that when using the norm loss $||\text{sign}(h) - h||_2$ to optimize the hidden code, the auto-encoder became very hard to be optimized, and the generated codes quickly collapse to some point. Thus in some previous works, they also add extra regulation terms to prevent the collapse, such as the orthogonality regulation on projection matrix [13]. By contrast, our model alleviates this optimization difficulty by the two discriminators, which is more flexible and achieves better results.

5 ANALYSIS

5.1 The Advantage of AE-GAN

In this subsection, we try to prove that the proposed AE-GAN could reconstruct the input with better fidelity. However, evaluating the quality of synthesized images is an open difficult problem [40]. In the auto-encoder setting, the output lies on continues feature space, so the likelihood of the generated image is difficult to calculate. Traditional metrics such as per-pixel Euclidean distance do not assess joint statistics of the result, and therefore do not measure the very structure and information that we want to capture in the input.

Some recent works [1, 12, 36] have tried to use pre-trained image semantic classifiers to measure the quality of the generated stimuli as a pseudo-metric. The intuition is that if the generated images are realistic, classifiers trained on real images will be able to identify
the synthesized image correctly as well. Similar to them, we adopt a popular Inception\textsuperscript{7} \cite{inception} model. For the input \( x \), the output of the Inception could be denoted as \( p(y|x) \) that corresponding to the distributions of the input categories, and \( p(y|x) \) is the output distribution of generated image \( \hat{x} \) going through the Inception model. Then we can use the divergence between \( p(y|x) \) and \( p(\hat{y}|\hat{x}) \) to evaluate the quality of the reconstructed image, which is referred as the Inception score:

\[
\text{Inception Score} = \mathbb{E}_x (KL(p(y|x)\|p(\hat{y}|\hat{x}))) = \sum_x \sum_y p(y|x) \log \frac{p(y|x)}{p(\hat{y}|\hat{x})} \quad (14)
\]

Thus if the reconstructed image is more similar to the input, the Inception score will be lower. For this comparison, we conduct the experiment on ILSVRC2012 dataset with 1,000 categories. We select two baselines for comparison, namely L1 and L2. L1 minimize the reconstruction error by \( |x - \hat{x}| \) and L2 minimize the reconstruction error by Equation 3. The Inception score compared with different auto-encoder objective is shown in Figure 7.

It could be seen from the figure that AE-GAN performs better than the L1 or L2 loss. Besides, when the AE-GAN is pre-trained with traditional L2 objective (L2+AE-GAN), our model obtains the best Inception score. We think the reason is: the objective of the auto-encoder is to encode the information of the input, however, the objective of traditional L1 or L2 norm loss is to minimize the summation of per-pixel feature distance, which results in a reconstructed image that looks similar with the input only at a global view. In AE-GAN the distance metric is measured by a deep neural network (discriminator). Thus it is flexible to capture both global and local information of the input \cite{autoencoder, gan}. We plot some examples in Figure 8, and we can see that the reconstruction by L1 or L2 loses a lot of details. By contrast, in the AE-GAN the reconstruction is more clear.

5.2 Advantage of B-GAN

Another contribution of the proposed methods is to approximate the binarization by introducing the adversarial objective \( \mathcal{L}_G \). On the contrary, most previous methods \cite{B-GAN, GAN, AE-GAN} apply the norm constraint on the hidden codes to make it approximate to binary:

\[
\mathcal{L}_b = ||\text{sign}(\mathbf{H}) - \mathbf{H}||_2 \quad (15)
\]

where \( \mathbf{H} \) is the codes and \( \text{sign} \) function maps the input to \( \pm 1 \).

In order to compare \( \mathcal{L}_G \) and \( \mathcal{L}_b \) quantitatively, inspired by Precision-Recall curve in the evaluation of traditional information retrieval tasks, we proposed the Fidelity-Binarization (F-B) curves. The fidelity could be the similarity between the input and output, and the binarization could be regarded as the distance between the hidden codes and its binary vector. The two metrics are opponent just like the precision and recall: If we want to increase the fidelity, we must relax the binary constraint on the hidden codes and vice versa. Fidelity could be calculated by the \( L_2 \) distance between reconstruction and the input. To evaluate the binarization property, we consider three aspects of the hidden codes:

\[
\begin{align*}
    b_1 &= -\log\left(\frac{1}{nb} \sum_{i=1}^{n} \sum_{j=1}^{b} (1 - |H_{ij}|)^2 \right) \\
    b_2 &= -\log\left(\frac{1}{nb} \text{trace}(H^T \cdot H) \right) \\
    b_3 &= -\log\left(\frac{1}{nb} |H^T \cdot 1| \right)
\end{align*}
\]

where \( n \) is the data size and \( b \) is the code length. The \( b_1 \) considers hidden codes approximation to binary. \( b_2 \) considers the independence of the hidden codes, and the \( b_3 \) considers the balance of the codes, which is equal to the proportion of the hidden codes to be fired. These three metrics has been adopted as the training criteria in some previous works \cite{B-GAN, GAN}. As the elements of \( \mathbf{H} \) lies in \((-1, 1)\), there exist upper bound for the three metrics. We use this upper bound to normalize the three term to restrict each of them to \([0, 1] \) \((b_1', b_2', b_3')\), and then combine this three metrics linearly to form the quality of the binarization.

\[
\text{binarization} = b_1' + b_2' + b_3'
\]

As the only focus of this section is to measure the improvement of B-GAN, we adopt the \( L_2 \) norm loss as the reconstruction objective.

---

\textsuperscript{7}We use the off-the-shelf public Inception-v3 model from https://download.pytorch.org/models/inception_v3_google-1a9a5a14.pth.
Figure 9: F-B curve for different bits on CIFAR10. The X-axis is the reconstruction value that has been normalized to $[0, 1]$. \( \beta = 0 \) means we only optimize for binary loss, \( \beta = 1 \) means we only optimize the reconstruction loss. \( L_2 \) refers to Equation 15 and \( L_1 \) replace the Euclidean norm with \( L_1 \) distance.

Thus the overall object could be formulated as:

\[
\mathcal{L} = \beta \cdot \mathcal{L}_{\text{rec}} + (1 - \beta) \cdot \mathcal{L}_{\text{binary}} \tag{18}
\]

where \( \mathcal{L}_{\text{binary}} \) denotes \( \mathcal{L}_G \) or \( \mathcal{L}_h \), \( \beta \) is the weight of the two objective. When we set \( \beta \) to 1, we only minimize the reconstruction error, and setting it to 0 means we only optimize the binarization. Tuning this weight we can plot the binarization curve at different fidelity value. We use the CIFAR10 dataset and tuning the hidden codes size from 32 to 256, the results are shown in Figure 9.

It could be seen from the figure that our proposed model could achieve a better result than traditional \( L_1 \) or \( L_2 \) loss. To minimize the reconstruction error, the model with norm constraints has to sacrifice the binarization to obtain good fidelity. As a result, when we increase the weight of fidelity, the binarization curves drop quickly. For the proposed model, the binarization error stays relatively stable, which means our model could reduce the reconstruction error and binarization error simultaneously and efficiently. The advantage of our model is more significant when the code length increase. In the experiment, we found that the gradient of the \( L_1 \) or \( L_2 \) became unstable with the code length increasing, and requires a meticulous hyper-parameter calibration. By contrast, in GMANs we relax the norm constraint with the GAN and the binarization is achieved by reducing discriminator error, so it is relatively insensitive to the code length. This demonstrates the advantage of applying adversarial training for hidden codes generation.

5.3 GMANs on Supervised Setting

The proposed model is based on unsupervised setting, and it could be trivially extended to the supervised scenario. In the supervised setting, not only the raw-sample but also the corresponding labels are given, such as the data category, pairwise distance etc. Image similarity in the original feature space may not fully reflect the semantic relationship between them, and it has been revealed that the extra labels could enrich the semantic capacity of the auto-encoder based hashing model [3, 4, 10].

In this paper, suppose the label or tag information associated with the input \( x \) is \( y \), which is a one-hot vector indicating the class of the input. Similar with previous works on semantic hashing that utilizing the label information [4, 10], we build a classifier on the hidden representation \( h \). The classifier is based on MLP and we adopt the cross-entropy to optimize the classifier:

\[
\tilde{h} = \text{MLP}(h) \\
\tilde{y} = \text{Softmax}(h) \\
\mathcal{L}_S = -\sum_{j=1}^{C} y_j \cdot \log \tilde{y}_j \tag{19}
\]

where \( C \) is the number of classes. Adding this term to Equation 13 we obtain the supervised objective of the generator:

\[
\mathcal{L}_{G+s} = \lambda_1 \cdot \mathcal{L}_{G}^{AE} + \lambda_2 \cdot \mathcal{L}_h^2 + \lambda_3 \cdot \mathcal{L}_V + \lambda_4 \cdot \mathcal{L}_S \tag{20}
\]

where \( \lambda_4 \) is the parameter to tune the weight of the classifier.

Similar to the unsupervised setting, we apply the supervised GMANs (SGMANs) on two images and two texts datasets: 1) CIFAR10; 2) MNIST; 3) RCV1; 4) 20Newsgroups. For the supervised weight \( \lambda_4 \) in Equation 20, we gradually increase it from 0 to 1, which means we first learn a good auto-encoder and then embed the label supervision into the learning process. For all setups, the classifier is a two-layer MLP with the hidden size equal to 64. We adopt four supervised hashing models for comparison. 1) Supervised Discrete Hashing (SDH) [41]. 2) Binary Reconstructive Embedding (BRE) [29]. 3) BDNN [10]. 4) Supervised VDSH (SVDSH) [4]. For all datasets, we evaluate the SGMANs by precision@2. The result is shown in Table 3.

We can see from table 3 that similar with the unsupervised application, our proposed GMANs also excels on supervised setting. The extra label information is embedded to the hashing process, and dynamically tuning the training process of hidden representation. Compared to unsupervised result in table 1 and 2, the supervised

<table>
<thead>
<tr>
<th>d</th>
<th>CIFAR10</th>
<th>MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>SDH [41]</td>
<td>31.60</td>
<td>62.23</td>
</tr>
<tr>
<td>BRE [29]</td>
<td>23.84</td>
<td>41.11</td>
</tr>
<tr>
<td>BDNN [10]</td>
<td>54.12</td>
<td>67.32</td>
</tr>
<tr>
<td>SVDSH [4]</td>
<td>45.41</td>
<td>59.59</td>
</tr>
<tr>
<td>SGMANs</td>
<td>58.48</td>
<td>69.20</td>
</tr>
</tbody>
</table>

Table 3: Supervised results of Precision at hamming distance less than 2.
model could obtain higher precision on the retrieval task, which confirms the previous observation that the semantic information is important for hashing [3, 4, 10]. In addition, different from previous methods that using $L_1$ or $L_2$ norm constraint, we use two GANs to optimize the generated codes to make it as close as possible to binary and conserves the input information. Furthermore, we proposed a variance control method to eliminate the mode collapse problem and design a self-controlled mechanism to stabilize the training process of GANs. Several supervised and unsupervised experiments in text and image datasets reveal the advantage of the proposed GMANs. In addition, we do a comprehensive analysis to show the advantage of the two adversarial training process of the proposed model. In the future, we want to extend the GMANs to other hashing models rather than auto-encoder to improve the generated codes.

7 ACKNOWLEDGEMENTS

The research work was supported by the National Science Foundation of China (No. 61533018, 61502493). This work was also supported by a grant from Ant Financial Services Group and supported by the independent research project of National Laboratory of Pattern Recognition.

REFERENCES

[10] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Manghistic, and Aaron Courville. 2016. ADVERSARIAL LEARNING. In LECV.