Unsupervised Deep Generative Adversarial Hashing Network

Kamran Ghasedi Dizaji[1], Feng Zheng[1], Najmeh Sadoughi Nourabadi[1], Yanhua Yang[2]
Cheng Deng[2], Heng Huang[1*]
[1]Electrical and Computer Engineering Department, University of Pittsburgh, PA, USA
[2]Xidian University, Xi'an, Shanxi, China
kag221@pitt.edu, feng.zheng@pitt.edu, najme.sadoughi@gmail.com, yanhyang@xidian.edu.cn
chdeng.xd@gmail.com, heng.huang@pitt.edu

Abstract

Unsupervised deep hash functions have not shown satisfactory improvements against their shallow alternatives, and usually require supervised pretraining to avoid overfitting. In this paper, we propose a new deep unsupervised hashing function, called HashGAN*, which efficiently obtains binary representation of input images without any supervised pretraining.* HashGAN *consists of three networks, a generator, a discriminator and an encoder. By sharing the parameters of the encoder and discriminator, we benefit from the adversarial loss as a data-dependent regularization in training our deep hash function. Moreover, a novel hashing loss function is introduced for real images, which results in minimum entropy, uniform frequency, consistent and independent hash bits. Furthermore, we employ a collaborative loss in training our model, enforcing similar random inputs and hash bits for synthesized images. In our experiments,* HashGAN *outperforms the previous unsupervised hash functions in image retrieval and achieves the state-of-the-art performance in image clustering on benchmark datasets. We also provide an ablation study, showing the contribution of each component in our loss function.*

1. Introduction

Image similarity search in big datasets has gained tremendous attentions in different applications such as information retrieval, data mining and pattern recognition [47]. With rapid growth of image data, it has become crucial to find compact and discriminative representations of images in huge datasets in order to have efficient storage and real-time matching for millions of images. Hashing functions provide an effective solution for this problem by attributing a binary code to each image, and consequently reducing the similarity search between high dimensional images to calculating the Hamming distance between their binary codes [14, 50, 33, 30]. Typically, hash functions are carefully designed to extract distinctive patterns from images relevant to their semantic categorizes, while being robust to various image transformations such as rotation, translation, scale, and lightning [32, 55, 22].

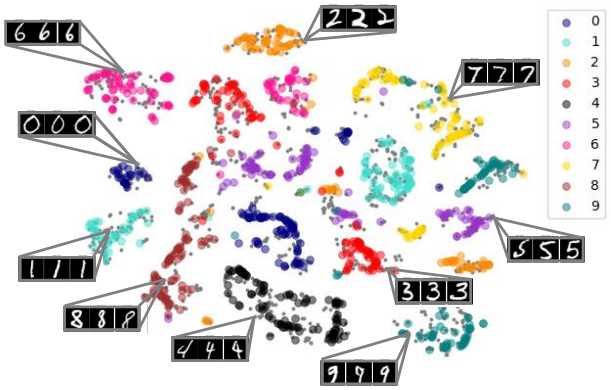


Figure 1: Visualization of *HashGAN* discriminative representations for a query set on *MNIST* using *TSNE* projection. The real and synthesized data are indicated by colored and gray circles respectively. Some of the synthesized images are randomly shown from different parts of space.

Generally, hash functions can be divided into supervised [33, 52, 16, 30] and unsupervised methods [18, 50, 46, 19]. The supervised hashing methods, especially deep hash functions [27, 8, 59, 55], showed remarkable performance in representing input data with binary codes. Although, these deep hash functions take advantages of deep learning models in representing images with discriminative attributes, they require costly human-annotated labels to train their large set of parameters. Thus, their performance is dramatically degraded by getting stuck in bad local minima when there is not enough labeled data for training.

*Corresponding Author. KD, FZ, HH were supported by NSF IIS 1302675, IIS 1344152, DBI 1356628, IIS 1619308, IIS 1633753, NIH R01 AG049371.

The unsupervised hashing methods address this issue by providing learning frameworks without requiring any supervisory signals. The unsupervised hashing methods either use shallow models with hand-crafted features [3, 39, 29, 2] as inputs, or employ deep architectures for obtaining both discriminative features and binary hash codes together. However, the unsupervised shallow functions may not capture the non-linear similarities between real-world images due to their low capacity. They also suffer from hand-crafted features and dimension reductions techniques (e.g. principle component analysis (PCA)), which are not robust to noise and image transformations. On the other hand, the unsupervised deep hash functions usually have insignificant improvements against the shallow models, since they can not exploit the power of deep models due to overfitting and lack of supervision. Some of the unsupervised deep hash functions tackle this issue by initializing their parameters using supervised pretraining with large datasets (*e.g.* ImageNet dataset [6]) [32, 22].

We propose a new unsupervised deep hashing model, called *HashGAN*, which nor suffers from shallow hash functions and hand-crafted features, neither needs the supervised pretraining to have discriminative binary codes. Our framework jointly learns a hash function with a generative adversarial network (*GAN*). In particular, we tie the discriminator of the *GAN* with the hash function, employing the adversarial loss function as a data-dependent regularization term in training our deep hash function. Furthermore, we introduce a novel loss function for hashing real images, minimizing the entropy of hash bits for each image, maximizing the entropy of frequency of hash bits, improving the consistency of hash codes against different image transformations, and providing independent hash bits. Moreover, we provide a collaborative loss function, which enforces the encoder to have the same binary hash code for a synthesized image by the generator, as the binary input variable provided to the generator while synthesizing the image. We show that this collaborative loss function is a helpful auxiliary task for obtaining discriminative hash codes.

Figure 1 illustrates a 2D visualization of *HashGAN* hash codes for a query set of real and fake images on *MNIST* dataset [28]. As shown, *HashGAN* not only achieves discriminative representations for real data, but also generates synthesized images conditioned on their binary inputs, representing the semantic categories. Experimental results indicate that our proposed model outperforms unsupervised hash functions with significant margin in information retrieval tasks. Moreover, *HashGAN* achieves superior or competitive results compared to the state-of-the-art models in image clustering tasks. We also explore the effect of each term in our loss function using an ablation study. Therefore, our experiments confirm the effectiveness of *HashGAN* in unsupervised attribute learning across different tasks. Our contributions are summarized as follows:

- Proposing a novel framework for unsupervised hashing model by coupling a deep hash function and a generative adversarial network.
- Introducing a new hashing objective for real images, regularized by the adversarial and collaborative loss functions on synthesized images, resulting in minimum-entropy, uniform frequency, consistent, and independent hash bits.
- Achieving state-of-the-art results compared to alternatives on information retrieval and clustering tasks.

2. Related Works

2.1. Hash Functions

Generally, hash functions can be grouped into supervised [33, 11, 52, 31, 16] and unsupervised methods [18, 50, 46, 19]. The supervised methods require class labels or pairwise similarity ground truths in their learning process, whereas the unsupervised approaches need only input samples. With the growing success of deep learning in different applications, several studies have been published about supervised deep hash functions [27, 8, 59, 30, 55]. They mostly use pairwise relationships in different variants of ranking loss functions (*e.g.* triplet [49], contrastive [17] objectives) to simultaneously learn discriminative features and encode hash bits. However, the performance of these supervised hashing models crucially depends on availability of labeled data in the training process.

Among the shallow models, locality sensitivity hashing (*LSH*) [13] maps original data into a low dimensional feature space using random linear projections, and then obtains binary hash codes. Later in [26, 51], *LSH* was extended to kernel-based variants of hash functions. Gong et al. introduced another well-known model, called iterative quantization (*ITQ*) [14], which uses an alternating optimization approach for learning efficient projections and performing binarization. Spectral hashing (*SpeH*) [50] computes binary hash codes by implementing spectral graph partitioning using the similarity information in a feature space. However, these models suffer from shallow hash functions and inflexible hand-crafted features, which limit their capabilities in dealing with complex and high dimensional real world data.

In unsupervised deep hashing models, semantic hashing [40] is one of the early studies, which adopts Restricted Boltzmann Machine (*RBM*) [20] model as a deep hash function, and trains its parameters using an unsupervised learning approach. Deep Hashing (*DH*) [10] applies an unsupervised loss function to a hierarchical neural networks to have quantized, balanced and independent hash code bits. Lin *et al.* introduced *DeepBit* [32] as an unsupervised deep hashing algorithm by defining an objective function based on quantization loss and balanced and rotation invariant hash

bits. In addition to quantization and balanced hash bits loss functions, unsupervised triplet hashing (*UTH*) [22] employs an unsupervised triplet loss, which minimizes the distance of an anchor image and its rotated version (*i.e.* positive pair) while maximizing the distance of the anchor image with a random image (*i.e.* negative pair). Another method with two steps is introduced in [21] to learn discriminative binary representations in an unsupervised manner. A convolutional neural network (*CNN*) is trained using a clustering algorithm in the first step, and then the learned cluster assignments are used as soft pseudo labels in a triplet ranking loss for training a deep hash function in the second step.

Our proposed model falls in the category of unsupervised deep hash functions. But unlike the unsupervised deep hash functions, which have insignificant improvements over the shallow alternatives, and/or require supervised pretraining using a large labeled dataset, *HashGAN* outperforms unsupervised alternatives with significant margins without any supervised pretraining.

2.2. Applications of GAN

Goodfellow *et al.* proposed a powerful generative model, called generative adversarial networks (*GAN*) [15], which is able to synthesize realistic images with great details. Particularly, *GAN* objective includes a two-player minimax game between two networks, a generator and a discriminator. The discriminator aims to distinguish between the real and synthesized (*i.e.* fake) images, and the generator maps samples from arbitrary distribution (*i.e.* random noise) to the distribution of real images, trying to synthesize fake images that fool the discriminator. Several studies [7, 38] further addressed problems such as the unstable training process of *GAN* and noisy and blurry synthesized images, resulting in higher quality images. Moreover, some works [35, 37] tried to improve the quality and diversity of generated images by conditioning on the supervisory signals like class labels and text descriptions, and incorporating these supervised information into the generative and discriminative pathways. In addition, *GAN* has been adopted in supervised and semi-supervised tasks to use the input data distribution as a generalization force, and enhance the classification results [44, 41, 38]. Unlike these supervised/semi-supervised studies, our model employs *GAN* in the unsupervised hashing task, and does not require any supervisory signals like class labels and image captions.

3. Unsupervised Deep Generative Adversarial Hashing Network

In this section, we first introduce *HashGAN* by showing its architecture and explaining the intuition behind the model. Then, we define its loss function and describe the effect of each term in our learning framework.

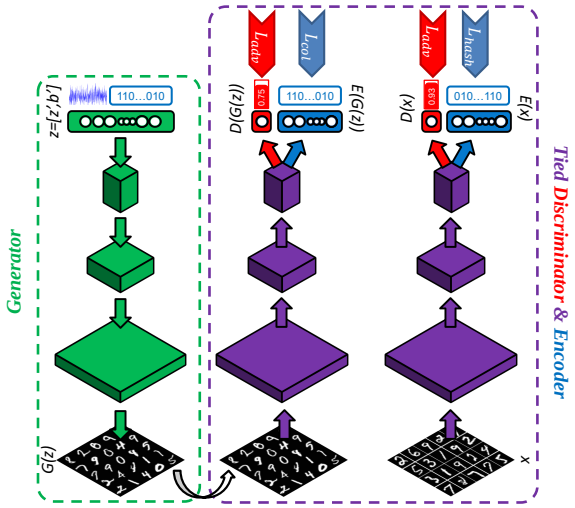


Figure 2: *HashGAN* architecture, including a generator (green), a discriminator (red) and an encoder (blue), where the last two share their parameters in several layers (red⊕blue=purple). The arrows on top represent the loss functions.

3.1. HashGAN Architecture

Our proposed *HashGAN* model consists of three components, a generator, a discriminator and an encoder. The generator is supposed to synthesize images that fool the discriminator by mapping samples from a random distribution to the real data distribution. The discriminator is expected to distinguish the synthesized images from real ones. The encoder is designed to map the images to discriminative binary hash codes. As shown in Figure 2, the discriminator and encoder share all of their parameters except for the weights of their last layer. The inputs of generator are also the concatenation of samples from two random distributions, including binary and uniform random variables.

In order to train the discriminator parameters, we use the standard adversarial loss function in *GAN* models. The parameters of encoder are trained via a hashing loss on real data and an ℓ_2-norm loss on fake data. The hashing loss ensures having quantized, balanced, consistent and independent hash codes for real images, and the ℓ_2-norm loss is determined to have similar hash codes as the generator binary inputs for synthesized images. To train the parameters of generator, we utilize the feature matching loss, introduced in [41], to match the statistics of the real and fake images. To do so, the expected value of the features in the last hidden layer of discriminator (encoder) network is selected in the feature matching loss function.

HashGAN architecture has several advantages in our unsupervised deep learning framework. First, tying the discriminator and encoder is very useful in unsupervised train-

ing of our deep hash function, because the adversarial loss can be considered as a data-dependent regularization term in training *HashGAN*, which avoids overfitting and getting stuck in bad local minima. From another point of view, the encoder pathway utilizes the information in the data distribution, which is discovered in the latent variables of discriminator.

It has been shown that interpolations in the input space of the generator produce semantic variations along data distribution [38, 9]. Hence, training the encoder to utilize these information hidden in the input variables of generator is helpful in learning discriminative binary codes. The feature matching loss and the ℓ_2-norm loss for training the generator and encoder networks can be considered as collaborative loss functions, which aim to use the generator binary inputs as the pseudo-hash-labels for the synthesized images, while they have similar statistics with the real images. This novel approach fits our unsupervised hashing problem, and it is different with the conventional conditional *GAN* models [35, 37], which need supervisory signals.

3.2. HashGAN Loss Function

Consider there are N images in the gallery set, denoted by $\mathbf{X} = \{\mathbf{x}_i | i = 1, \cdots, N\}$, which are used in training our deep hash function. We utilize a multi-layer hash encoder to map the input images into the K-bit hash codes. To do so, there are K independent sigmoid functions in the last layer of our encoder network. Thus, the output of encoder for each image is represented by $\mathbf{t}_i = \mathcal{E}(\mathbf{x}_i)$, which shows the composition of K independent probabilities as $t_{ik} = P(b_{ik} = 1|\mathbf{x}_i; \mathbf{W}_\mathcal{E})$, where t_{ik} and b_{ik} are the k-th output of encoder and binary hash code for the i-th image, and $\mathbf{W}_\mathcal{E}$ indicates the encoder parameters. Note that the binary hash codes are simply computed using $b_{ik} = \mathbf{1}(t_{ik} > 0.5)$, where $\mathbf{1}(.)$ is the indicator function.

Our *HashGAN* model employs a generator network, which maps the samples from a random distribution to the data distribution. As mentioned earlier, the random input of generator is concatenation of binary and uniform random variable as $\mathbf{z}_i = [\mathbf{z}'_i, \mathbf{b}'_i]$, where $\mathbf{z}'_i \sim \mathcal{U}(0, 1)$ shows the uniform random noise and $\mathbf{b}'_i \sim \mathcal{B}$ indicates the binary random noise. While the real images are shown by $\mathbf{x}_i$, the synthesized images by the generator are represented by $\hat{\mathbf{x}}_i = \mathcal{G}(\mathbf{z}_i)$. We also obtain the encoder outputs for the synthesized images as $\hat{\mathbf{t}}_i = \mathcal{E}(\hat{\mathbf{x}}_i) = \mathcal{E}(\mathcal{G}(\mathbf{z}_i))$.

The discriminator of *HashGAN* is supposed to determine whether its input image is a real or a synthesized sample. A sigmoid function is considered as the last layer of discriminator, computing the probabilities $p_i = \mathcal{D}(\mathbf{x}_i) = P(y_i = 1|\mathbf{x}_i; \mathbf{W}_\mathcal{D})$ and $\hat{p}_i = \mathcal{D}(\mathcal{G}(\mathbf{z}_i)) = P(y_i = 1|\hat{\mathbf{x}}_i; \mathbf{W}_\mathcal{D})$, where p_i and $\hat{p}_i$ are the probabilities of being real ($y_i = 1$) for the i-th real and synthesized images respectively.

Now, we are able to define the loss function in our learning framework. The total loss function is summation of the adversarial loss, hashing loss, and collaborative loss for the real and synthesized images:

$$\mathcal{L}_{total} = \mathcal{L}_{adv} + \mathcal{L}_{hash} + \mathcal{L}_{col}\,. \tag{1}$$

Following, we describe each term of the loss function in more details, and explain the role of each one in achieving discriminative binary hash codes. As proposed in [15], the adversarial loss in *GAN* models is designed as a minimax play between the discriminator and the generator models, in which the discriminator is trained to correctly distinguish the real and synthesized images, and the generator is trained to synthesize fake images that fool the discriminator. The adversarial loss function for training our discriminator has the following form:

$$\max_{\mathcal{D}} \;\; \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log(\mathcal{D}(\mathbf{x}))\big] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\big[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}))\big] \tag{2}$$

where the goal is to train the discriminator $\mathcal{D}$ to distinguish the real image $\mathbf{x}$ from the synthesized sample $\mathcal{G}(\mathbf{z})$. The adversarial loss is maximized *w.r.t.* the discriminator to increase the log-likelihood of correct predictions on real images and decrease the log-likelihood of mis-prediction on fake samples.

The hashing objective for real data contains four losses, including minimum-entropy, uniform frequency, consistent, and independent bits loss functions. The following equation shows these loss functions:

$$\begin{aligned}\min_{\mathcal{E}} \;\; & \underbrace{-\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log t_{ik} + (1 - t_{ik})\log(1 - t_{ik})}_{\text{minimum entropy bits}} \\ & + \underbrace{\sum_{k=1}^{K} f_k \log f_k + (1 - f_k)\log(1 - f_k)}_{\text{uniform frequency bits}} \\ & + \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} \|t_{ik} - \tilde{t}_{ik}\|_2^2}_{\text{consistent bits}} + \underbrace{\|\mathbf{W}_\mathcal{E}^{L\top}\mathbf{W}_\mathcal{E}^{L} - \mathbf{I}\|_2^2}_{\text{independent bits}}\,,\end{aligned} \tag{3}$$

where $\tilde{t}_{ik} = P(b_{ik}|\tilde{\mathbf{x}}_i; \mathbf{W}_\mathcal{E})$ is the k-th encoder output for the i-th real image, transformed by translation, rotation, flipping, or noise, $f_k = 1/N \sum_{i=1}^{N} t_{ik}$ is the frequency of the k-th hash bit code over sampled images, and $\mathbf{W}_\mathcal{E}^L$ is the weights of the last layer on the encoder network.

The first term in the hashing loss function is equivalent to entropy of each hash bit, and minimizing this term pushes hash bits for each image toward 0 or 1. Thus, the minimum-entropy bits loss function reduces the quantization loss without using the sign function. Considering f_k

as the empirical frequency of each hash bits, the second term in this loss function is a negative of entropy for the bits frequency. By maximizing (i.e. minimizing negative of) the entropy of bits frequency, the encoder tends to have balanced hash codes. The third term in the loss function constrains the encoder to extract similar hash codes for an image and its transformed variants, making the encoder robust to the transformations. Finally, the last term in this loss function pushes the encoder to have independent hash bits.

We also take advantages of the synthesized images in training the encoder network by a ℓ_2-norm loss function, which minimizes the distance of encoder outputs and generator binary inputs. Following equation shows the ℓ_2-norm loss on the synthesized data:

$$\min_{\mathcal{E}} \ \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\left[\|\mathcal{E}(\mathcal{G}(\mathbf{z})) - \mathbf{b}'\|_2^2\right], \tag{4}$$

where $\mathbf{b}'$ is the binary random variable in the generator input $\mathbf{z} = [\mathbf{z}', \mathbf{b}']$. Using this ℓ_2-norm loss function, the encoder network is able to provide similar hash codes for the synthesized images, which share the same binary attributes $\mathbf{b}'$, but vary due to different uniform random variables $\mathbf{z}'$.

In order to train the generator network, we used the feature matching loss instead of directly optimizing the output of the discriminator via the traditional adversarial loss function. The feature matching loss requires the generator to synthesize images that have similar statistic to the real images. We consider the last hidden layer of discriminator, denoted by $\mathcal{F}$, as the source of statistic, and define the following loss function:

$$\min_{\mathcal{G}} \ \|\mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\mathcal{F}(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\mathcal{F}(\mathcal{G}(\mathbf{z}))\|_2^2, \tag{5}$$

where $\mathcal{F}$ is also the last hidden layer of encoder network, affecting the hash codes and the adversarial probability. The feature matching loss provides more stability in training our model, and leads the synthesized images to share statistic with real data. This is very helpful in collaborating with ℓ_2-norm loss, making the pseudo-hash-labels for fake data effective on obtaining discriminative binary representations for real images.

In order to train our *HashGAN* model, we are able to use stochastic learning techniques. Thus, we alternatively train the generator and tie the discriminator and the encoder networks. In particular, we optimize the parameters of discriminator and encoder jointly using the adversarial, hashing and ℓ_2-norm loss functions in one step, and train the parameters of generator using the feature matching loss in the next step.

4. Experiments

We perform several experiments to evaluate the performance of our proposed model on multiple datasets. The quality of hash codes extracted by *HashGAN* is explored in image retrieval and clustering tasks. We also investigate the effect of each component in our loss function using an ablation study.

Implementation details: We use almost similar architectures for *HashGAN* to the *Improved-GAN* networks in [41]. We avoid pooling layers and use strided convolutional layers, utilize weight normalization [42] to stabilize the training process, consider ReLU and leaky-ReLU nonlinearities [34] as the activation function of convolutional layers in our discriminator and encoder. For image preprocessing, we only normalize the image intensities to be in the range of $[0, 1]$ or $[-1, 1]$, and consequently use sigmoid and TanH functions in the last layer of our generator. A zero mean Gaussian noise with standard deviation of 0.15 is also added to the input images of our discriminator/encoder. Moreover, we set the learning rate to 9×10^{-4} and linearly decrease it to 3×10^{-4}, and adopt Adam [23] as our optimization method with the hyperparameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e-08$. Since our hashing task is unsupervised, we did not tune any hyperparameters for adjusting the effect of our losses in different datasets, and use the default setting. In particular, we set the weights for the adversarial ($\mathcal{L}_{adv}$), feature matching ($\mathcal{L}_{feat}$), independent bits ($\mathcal{L}_{indBit}$), uniform frequency bits ($\mathcal{L}_{uniFrqBit}$), consistent bits ($\mathcal{L}_{consBit}$) loss functions equal to 1, and the weight of ℓ_2-norm loss ($\mathcal{L}_2$) equal to 0.1. For $\mathcal{L}_{minEntrpBit}$ in the hash loss function, the weight is selected from $\lambda_{minEntrpBit} = \{10^{-3}, 10^{-2}\}$ based on the final epoch loss value in the training process. Besides, we first train *HashGAN* without the hash and ℓ_2-norm loss functions by setting its weight equal to zero for one tenth of the maximum epoch, since the obtained hash codes in the first iterations may not be reliable for training the encoder parameters. We use Theano toolbox [1] for writing our code, and run the algorithm in a machine with one Titan X Pascal GPU.

Datasets: We compare our model with unsupervised hash functions in the image retrieval task on *CIFAR-10* [24] and *MNIST* [28]. Furthermore, we analyze the discriminative capability of *HashGAN* binary codes in the image clustering task on *MNIST*, *USPS*, *FRGC* [56] and *STL-10* [5] datasets. Following, we describe each dataset briefly.

CIFAR-10 dataset [24] contains 60K 32×32 colored images balanced across 10 classes (*i.e.* airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck).

MNIST dataset [28] includes 70K 28×28 gray scale images of hand written digits (0-9) across 10 classes.

USPS is a dataset of 11K 16×16 gray scale handwritten digits from USPS postal service, with unbalanced distribution across the ten digits.

FRGC contains $2,462$ facial images from randomly selected 20 subjects on this dataset [56]. Similar to [56], we crop the images to 32×32 colored facial images.

	Dataset	*CIFAR-10*						*MNIST*						Super. Pretrain
		mAP (%)			mAP@1000 (%)			mAP (%)			mAP@1000 (%)			
	Model	16	32	64	16	32	64	16	32	64	16	32	64	
Shallow	*KMH* [18]	13.59	13.93	14.46	24.08*	23.56*	25.19*	32.12	33.29	35.78	59.12*	70.32*	67.62*	✗
	SphH [19]	13.98	14.58	15.38	24.52*	24.16*	26.09*	25.81	30.77	34.75	52.97*	65.45*	65.45*	✗
	SpeH [50]	12.55	12.42	12.56	22.10*	21.79*	21.97*	26.64	25.72	24.10	59.72*	64.37*	67.60*	✗
	PCAH [46]	12.91	12.60	12.10	21.52*	21.62*	20.54*	27.33	24.85	21.47	60.98*	64.47*	63.31*	✗
	LSH [13]	12.55	13.76	15.07	12.63*	16.31*	18.00*	20.88	25.83	31.71	42.10*	50.45*	66.23*	✗
	ITQ [14]	15.67	16.20	16.64	26.71*	27.41*	28.93*	41.18	43.82	45.37	70.06*	76.86*	80.23*	✗
Deep	*DH* [10]	16.17	16.62	16.96	-	-	-	43.14	44.97	46.74	-	-	-	✗
	DAR [21]	16.82	17.01	17.21	-	-	-	-	-	-	-	-	-	✗
	DeepBit [32]	-	-	-	19.43	24.86	27.73	-	-	-	28.18	32.02	44.53	✓
	UTH [22]	-	-	-	28.66	30.66	32.41	-	-	-	43.15	46.58	49.88	✓
	HashGAN [ours]	**29.94**	**31.47**	**32.53**	**44.65**	**46.34**	**48.12**	**91.13**	**92.70**	**93.93**	**94.31**	**95.48**	**96.37**	✗

Table 1: Image retrieval results (mAP and mAP@1000) of unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 16, 32 and 64. The usage of supervised pretraining is shown for each model using the tick sign. The results of alternative models are reported from the reference papers, except for the ones marked by $(*)$ on top, which are obtained by us running the released code.

STL-10 database [5] includes 13K colored images across 10 classes (*i.e.* airplane, bird, car, cat, deer, dog, horse, monkey, ship and truck). The images are resized to 32×32.

4.1. Image Retrieval

Alternative models: For image retrieval, we compare our method with the previous unsupervised hash functions including K-means hashing (*KMH*) [18], spherical hashing (*SphH*) [19], spectral hashing (*SpeH*) [50], PCA-based hashing (*PCAH*) [46], locality sensitivity hashing (*LSH*) [13], iterative quantization (*ITQ*) [14], deep hashing (*DH*) [10], discriminative attributes representations (*DAR*) [21], *DeepBit* [32] and unsupervised triplet hashing (*UTH*) [22].

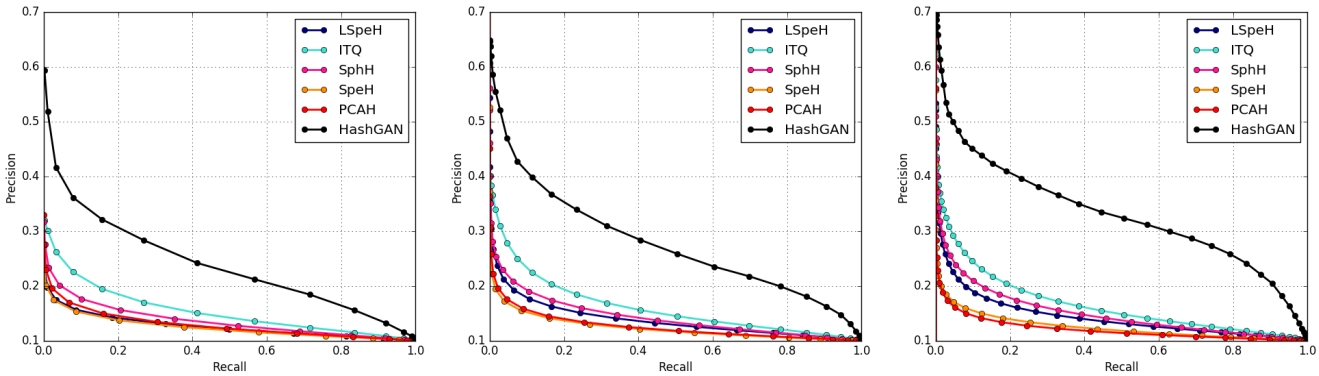
Evaluation metrics: We evaluate the performance of *HashGAN* compared to the aforementioned unsupervised hashing functions using precision and mean average precision (mAP). We follow the standard protocol for both *MNIST* and *CIFAR-10* datasets, and randomly sample 1000 images (100 per class) as the query set and use the remaining data as the gallery set. In particular, we report the results of the image retrieval in terms of precision@1000, mAP, and mAP@1000[1], where precision@1000 is the fraction of correctly retrieved samples from the top 1000 retrieved samples in gallery, mAP is the mean of the average precision of query images over all the relevant images, mAP@1000 is mAP calculated over the top 1000 ranked images from the gallery set. The reported results are the average of 5 experimental results.

Performance comparison: Table 1 shows the mAP and mAP@1000 results of *HashGAN* and other alternative models across different hash bit sizes. To better compare the models, we divide the hash functions into two groups of shallow and deep models, and indicate whether they use supervised pretraining or not. The results demonstrate that our model consistently outperforms other models with significant margins across different number of bits, datasets and metrics. Although, *HashGAN* gives better performance with more number of hash bits, its performance has small drops with less hash bits. Interestingly, the unsupervised deep hash functions, which use supervised pretraining via ImageNet dataset, show better results on *CIFAR-10* dataset compared to the shallow models, but have relatively lower performance on *MNIST* dataset. This shows that pretraining on ImageNet dataset is more helpful for *CIFAR-10* than for *MNIST*, which is not that surprising, given that ImageNet data distribution looks more similar to the *CIFAR-10* image distribution than *MNIST*. However, our model does not require any supervised pretraining, and consequently is not affected by pretraining biases, and achieves superior results on both datasets.

Table 2 indicates the results of precision@1000 for *HashGAN* and some of the unsupervised hash functions. Similar to Table 1, *HashGAN* achieves superior results in comparison with the alternative shallow and deep models. The improvements of our model are consistent across both *MNIST* and *CIFAR-10* datasets and different hash code sizes, showing the effectiveness of our learning framework in dealing with different conditions. We also compare *HashGAN* with the baselines using precision-recall curves on *CIFAR-10* dataset. Figure 3 clearly demonstrates better performance for *HashGAN* consistently across different number of bits.

Moreover, we visualize the *HashGAN*'s top 10 retrieved images for some query data on *CIFAR-10* dataset, when the

[1]Note that comparisons in some of the previous studies are confusing, as they comapre mAP results of baseline models with mAP@1000 results of other models. To avoid such confusion, we provide evaluations in terms of both of these metrics, separately.



(a) 16 bits (b) 32 bits (c) 64 bits

Figure 3: Precision-Recall curves on *CIFAR-10* database for *HashGAN* and five baselines with 16, 32, and 64 hash bits.

	Dataset	CIFAR-10			MNIST		
		precision@1000 (%)			precision@1000 (%)		
	Model	16	32	64	16	32	64
Shallow	*KMH* [18]	18.83	19.72	20.16	51.08*	53.82*	54.13*
	SphH [19]	18.90*	20.91*	23.25*	46.31*	54.74*	62.50*
	SpeH [50]	18.83	19.72	20.16	51.08*	53.75*	54.13*
	PCAH [46]	18.89	19.35	18.73	51.79*	51.90*	48.36*
	LSH [13]	16.21	19.10	22.25	31.95*	45.05*	55.31*
	ITQ [14]	22.46	25.30	27.09	61.94*	68.80*	71.00*
Deep	*DH* [10]	16.17	16.62	16.96	-	-	-
	DAR [21]	24.54	26.62	28.06	-	-	-
	HashGAN [ours]	**41.76**	**43.62**	**45.51**	**93.52**	**94.83**	**95.60**

Table 2: Image retrieval results (precision@1000) of unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 16, 32 and 64. The results of alternative models are reported from the reference papers, except for the ones marked by (∗) on top, which are obtained by us running the released code.

hash bit size is 32. Figure 4 illustrates these results, qualitatively showing that our hash function is able to extract semantic binary attributes.

4.2. Ablation Study

We perform an ablation study to examine the contribution of each loss component in the achieved results. We evaluate this experiment across L_{indBit}, L_2, $L_{consBit}$, $L_{uniFrqBit}$ and $L_{adv}+L_{feat}+L_2$. Note that in the absence of adversarial loss, the feature matching and ℓ_2-norm losses are also excluded due to their co-dependencies with the adversarial loss. We exclude loss components one at a time, measuring the difference in precision@1000 on *MNIST* and *CIFAR-10* datasets (See Fig. 5). The first observation is that all of the loss components contribute in improving the results. Furthermore, the figure shows the strong effect of $L_{adv} + L_{feat} + L_2$ as the key components in avoiding overfitting. In other words, employing GAN in our model has the highest practical contribution, and removing the discriminator and generator degrades the performance substantially. It also demonstrates that the presence of uniform frequency loss is very important. Examining the results achieved in the absence of this loss demonstrates that some of the binary codes collapse to either zero or one, reducing the capacity of the assigned hash bit size. The relative analysis of the results in each dataset demonstrates that consistency loss is more effective in *CIFAR-10* than in *MNIST*. This is expected as we only use noise for image transformation on MNIST since the images are centered and scaled, but rely on extra transformations including translations and horizontal flipping for *CIFAR-10*. The figure also demonstrates considerable contribution from the ℓ_2-norm loss, showing the effectiveness of our framework in using the synthesized images for training the encoder network. The lowest effect is provided by the independent bit loss.

Figure 4: Top 10 retrieved images for query data by *HashGAN* on *CIFAR-10* dataset with 32 bits hash code.

4.3. Image Clustering

One way to measure whether the hash function is effective in extracting distinctive codes is to evaluate their performance in clustering tasks. Hence, we assess *HashGAN*'s

	Dataset	*MNIST*		*USPS*		*FRGC*		*STL-10*	
	Model	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
Shallow	*K-means*	0.500	0.534	0.450	0.460	0.287	0.243	0.209*	0.284
	N-Cuts [43]	0.411	0.327	0.675	0.314	0.285	0.235	-	-
	SC-LS [4]	0.706	0.714	0.681	0.659	0.550	0.407	-	-
	AC-PIC [58]	0.017	0.115	0.840	0.855	0.415	0.320	-	-
	SEC [36]	0.779	0.804	0.511	0.544	-	-	0.245*	0.307
	LDMGI [57]	0.802	0.842	0.563	0.580	-	-	0.260*	0.331
Deep	*NMF-D* [45]	0.152	0.175	0.287	0.382	0.259	0.274	-	-
	DEC [53]	0.816	0.844	0.586	0.619	0.505	0.378	0.284*	0.359
	JULE-RC [56]	0.913	0.964	0.913	0.950	0.574	0.461	-	-
	DEPICT [12]	**0.917**	**0.965**	**0.927**	0.964	**0.610**	**0.470**	0.303*	0.371*
	HashGAN [ours]	0.913	**0.965**	0.920	0.958	0.602	0.465	**0.316**	**0.394**

Table 3: Clustering performance of *HashGAN* and several other algorithms on four image datasets based on accuracy (ACC) and normalized mutual information (NMI). The results of alternative models are reported from the reference papers, except for the ones marked by $(*)$ on top, which are obtained by us running the released code.

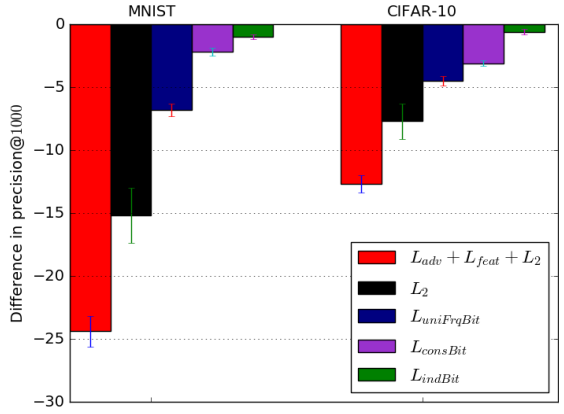


Figure 5: The difference in the precision@1000, when each of the loss components are excluded from the *HashGAN*'s objective function on *MNIST* and *CIFAR-10* datasets.

ability in clustering, by using the extracted hash codes as low dimensional input features for K-means and compare the results with alternative clustering models.

Alternative Models: We compare our clustering method with several baselines and state-of-the-art clustering algorithms, including *K-means*, normalized cuts (*N-Cuts*) [43], large-scale spectral clustering (*SC-LS*) [4], agglomerative clustering via path integral (*AC-PIC*) [58], spectral embedded clustering (*SEC*) [36], local discriminant models and global integration (*LDMGI*) [57], *NMF* with deep model (*NMF-D*) [45], task-specific clustering with deep model (*TSC-D*) [48], deep embedded clustering (*DEC*) [53], joint unsupervised learning (*JULE-RC*) [56] and *DEPICT* [12].

Evaluation metrics: To compare the clustering results of our model with previous studies, we rely on the two popular metrics used to evaluate clustering: *normalized mutual information* (NMI), and *accuracy* (ACC). NMI provides a measure of similarity between two data with the same label, which is normalized between 0 (lowest similarity) to 1 (highest similarity) [54]. To calculate ACC we find the best mapping between the predicted clusters and the true labels, following the approach proposed by [25].

Performance comparison: Table 3 gives the evaluation results for our clustering method and the mentioned algorithms in terms of NMI and ACC across *MNIST*, *USPS*, *FRGC*, and *STL-10* datasets. The results demonstrate that our method (*HashGAN* + K-means) achieves superior or competitive results compared to the state-of-the-art clustering algorithms. Note that our method is not specially designed for clustering, since we only run K-means algorithm on the *HashGAN* representations without backpropagating clustering error through the network. The table also indicates clear advantage of deep models compared with shallow models, emphasizing the importance of deep representations in image clustering. Overall, this experiment demonstrates the effectiveness of *HashGAN* model in extracting discriminative representations on different datasets in completely unsupervised manner.

5. Conclusion

This paper introduced *HashGAN*, an unsupervised deep hashing model, composed of a generator, a discriminator and an encoder. We defined a novel objective function to efficiently train our deep hash function without any supervision. Using the tied discriminator and encoder, we employed the adversarial loss as a data-dependent regularization for unsupervised learning of our hash function. Our novel hashing loss also led to quantized, balanced, consistent and independent hash bits for real images. Furthermore, we introduced a collaborative loss to use the synthesized images in training our hash function. *HashGAN* outperformed unsupervised hashing models in information retrieval with significant margin, and achieved state-of-the-art results in image clustering.

References

[1] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint*, 2016. 5

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE conference on Computer vision and pattern recognition (CVPR)*, pages 510–517. Ieee, 2012. 2

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *European conference on Computer Vision (ECCV)*, pages 778–792, 2010. 2

[4] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, 2011. 8

[5] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011. 5, 6

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009. 2

[7] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems (NIPS)*, pages 1486–1494, 2015. 3

[8] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision (ECCV)*, pages 219–234. Springer, 2016. 1, 2

[9] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016. 4

[10] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015. 2, 6, 7

[11] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *European Conference on Computer Vision*, pages 250–264. Springer, 2014. 2

[12] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, pages 5736–5745, 2017. 8

[13] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999. 2, 6, 7

[14] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *In Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 1, 2, 6, 7

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, 2014. 3, 4

[16] J. Guo, S. Zhang, and J. Li. Hash learning with convolutional neural networks for semantic based image retrieval. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 227–238. Springer, 2016. 1, 2

[17] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE computer society conference on Computer vision and pattern recognition*, volume 2, pages 1735–1742. IEEE, 2006. 2

[18] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013. 1, 2, 6, 7

[19] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012. 1, 2, 6, 7

[20] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 2

[21] C. Huang, C. Change Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5175–5184, 2016. 3, 6, 7

[22] S. Huang, Y. Xiong, Y. Zhang, and J. Wang. Unsupervised triplet hashing for fast image retrieval. *arXiv preprint arXiv:1702.08798*, 2017. 1, 2, 3, 6

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009. 5

[25] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955. 8

[26] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, 2012. 2

[27] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015. 1, 2

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2, 5

[29] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011. 2

[30] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1711–1717. AAAI Press, 2016. 1, 2

[31] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1963–1970, 2014. 2

[32] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1183–1192, 2016. 1, 2, 6

[33] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081. IEEE, 2012. 1, 2

[34] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, volume 30, 2013. 5

[35] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 3, 4

[36] F. Nie, Z. Zeng, I. W. Tsang, D. Xu, and C. Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011. 8

[37] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016. 3, 4

[38] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 3, 4

[39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *IEEE international conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011. 2

[40] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. 2

[41] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242, 2016. 3, 5

[42] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 901–909, 2016. 5

[43] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 8

[44] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015. 3

[45] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *International Conference on Machine Learning (ICML)*, pages 1692–1700, 2014. 8

[46] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010. 1, 2, 6, 7

[47] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014. 1

[48] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang. Learning a task-specific deep architecture for clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 369–377. SIAM, 2016. 8

[49] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 2

[50] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems (NIPS)*, pages 1753–1760, 2009. 1, 2, 6, 7

[51] H. Xia, P. Wu, S. C. Hoi, and R. Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 55–64. ACM, 2012. 2

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

[53] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016. 8

[54] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003. 8

[55] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 1, 2

[56] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. 5, 8

[57] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010. 8

[58] W. Zhang, D. Zhao, and X. Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013. 8

[59] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016. 1, 2