This ACCV 2022 paper, provided here by the Computer Vision Foundation, is the author-created version.
The content of this paper is identical to the content of the officially published ACCV 2022
LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/accv

Unified Energy-based Generative Network for Supervised Image Hashing

Khoa D. Doan[1,3], **Sarkhan Badirli**[2], **Chandan K. Reddy**[3]

[1]College of Engineering and Computer Science, VinUniversity
[2]Eli Lilly and Company, Indianapolis, IN, USA
[3]Department of Computer Science, Virginia Tech, Arlington, VA, USA
khoadoan106@gmail.com, s.badirli@gmail.com, reddy@cs.vt.edu

Abstract. Hashing methods often face critical efficiency challenges, such as generalization with limited labeled data, and robustness issues (such as changes in the data distribution and missing information in the input data) in real-world retrieval applications. However, it is non-trivial to learn a hash function in existing supervised hashing methods with both acceptable efficiency and robustness. In this paper, we explore a unified generative hashing model based on an explicit energy-based model (EBM) that exhibits a better generalization with limited labeled data, and better robustness against distributional changes and missing data. Unlike the previous implicit generative adversarial network (GAN) based hashing approaches, which suffer from several practical difficulties since they simultaneously train two networks (the generator and the discriminator), our approach only trains one single generative network with multiple objectives. Specifically, the proposed generative hashing model is a bottom-up multipurpose network that simultaneously represents the images from multiple perspectives, including explicit probability density, binary hash code, and category. Our model is easier to train than GAN-based approaches as it is based on finding the maximum likelihood of the density function. The proposed model also exhibits significant robustness toward out-of-distribution query data and is able to overcome missing data in both the training and testing phase with minimal retrieval performance degradation. Extensive experiments on several real-world datasets demonstrate superior results in which the proposed model achieves up to 5% improvement over the current state-of-the-art supervised hashing methods and exhibits a significant performance boost and robustness in both out-of-distribution retrieval and missing data scenarios.

Keywords: image hashing, generative energy-based models, retrieval

1 Introduction

Searching for similar items (such as images) is an important yet challenging problem in this digital world. Accurate retrieval within a constrained response time is crucial, especially in large databases with several millions of images. This motivates the need for approximate nearest-neighbor (ANN) methods instead of

using an intractable linear scan of all the images for such massive datasets. Hashing is a widely used ANN method with a principled retrieval approach for web-scale databases. In hashing, high-dimensional data points are projected onto a much smaller locality-preserving *binary* space. Searching for similar images is reduced to searching for similar discrete vectors in this binary space using computationally-efficient Hamming distance [1]. Searching for an item in the binary space is extremely fast because each Hamming distance calculation (e.g., for 64-bit vectors) only needs 2 CPU instructions in most modern hardware. Furthermore, the compact binary codes are storage-efficient, thus the entire index of items can be kept in fast-access memory; for example, a million 64-bit vectors only occupy approximately 8 megabytes. This paper focuses on the learning-to-hash methods that "learn" hash functions for efficient image retrieval.

The mapping between the original image x and the k-bit discrete vectors is expressed through a hash function $f : x \to \{-1, 1\}^k$. Learning and deploying such a hash function in real-world applications face many challenges. First, the hash function should capture the similarity relationship between images in the binary space, for example, represented in the annotated similarity between items. However, with a massive amount of data, the annotated similarity is scarce. This leads to a poor generalization in methods that exclusively rely on such annotated information. Furthermore, real-world data contains amendable missing information (e.g., a part of an image is corrupted during lossy compression or transmission between systems) and gradually changes over time (i.e., the underlying data distribution changes). A hash function that is not robust to such scenarios is not suitable for real-world applications because its expected retrieval performance will quickly degrade.

Several learning-to-hash methods, especially the supervised ones, have been proposed for efficient ANN search [2,3,4,5,6,7,8,9,10,11]. However, subject to the scarcity of similarity information, these methods run into problems such as overfitting and train/test distribution mismatch, resulting in a significant loss in retrieval performance. Recently, some methods employ generative models, specifically generative adversarial networks (GAN), to synthesize additional training data for improving the generalization of the learned hash functions. Nevertheless, these GAN-based methods do not take full advantage of generative models beyond synthetically generating the data. The main reason is that GAN is an implicit generative model that does not directly estimate the density function of the data. On the other hand, explicit generative models, specifically energy-based models (EBMs), can synthesize images and recover the missing information in those images through the inference of the EBMs. For example, when the EBM explicitly models the density of the data $p(x)$, we can recover the missing information in a data point x by revising x through the MCMC inference of the EBM to find the most probable version of x in the data distribution, thus effectively recovering x. Such ability of explicit generative models is extremely useful for real-world retrieval applications, especially when data loss or corruption can happen at any stage during the data collection or transmission process in these

applications. By recovering the corrupted data, we hope to preserve much of the retrieval performance of the model.

In this paper, we propose a unified energy-based generative hashing framework (GenHash) that simultaneously learns the representation of the images and the hash function. Our hashing network consists of a shared representation network. This network learns shared representation of the images that are useful to solve multiple objectives. Each objective is modeled as a lightweight head (a multi-layer perceptron) on top of the shared network and solves a specific task. The tasks include: 1) an explicit joint probability density estimation of an image and its semantic labels (energy head), 2) a contrastive hash-function learning (hash head) and 3) semantic label prediction (classification head). Consequently, this multipurpose network simultaneously learns to represent the images from multiple perspectives and allows the training process to develop a shared set of features as opposed to developing them redundantly in separate networks such as the GAN-based methods. Finally, since our model only trains the EBM for data synthesis, it requires fewer model parameters than approaches that use multiple networks (e.g., a generator and discriminator in GAN-based approaches). The main contributions of our paper are summarized below:

- We propose a unified generative, supervised learning-to-hash framework that takes complete advantage of generative energy-based models and enjoys better generalization and robustness towards missing data and-out-of distribution retrieval. The core component of this unified framework is the multi-headed or multipurpose hashing network, which combines density estimation (i.e., MCMC teaching process) and hash coding (i.e., contrastive loss).
- We propose a simple yet efficient training procedure to train the multipurpose hashing network. Specifically, we propagate the MCMC chains during training with two persistent contrastive divergence (PCD) buffers. One PCD buffer "explores" different modes of the model during training while the other PCD buffer is responsible for "exploiting" the learned modes to assist the contrastive hash function learning. The two PCD buffers jointly improve the efficiency of the MCMC teaching, thus allowing the training process to converge faster in practice.
- We demonstrate the advantages of our model over several state-of-the-art hashing techniques through an extensive set of experiments on various benchmark retrieval datasets.

The rest of the paper is organized as follows. We discuss the related work in Section 2. In Section 3, we describe the details of the proposed generative hashing network. We present quantitative and qualitative experimental results in Section 4 and conclude our discussion in Section 5.

2 Related Works

In this section, we review the previous research works related to two topics, namely, image hashing and energy-based generative models.

2.1 Image Hashing

Learning to hash, and especially image hashing, has been heavily investigated in both theory and practice. The existing image hashing methods can be organized into two categories: shallow hashing and deep hashing. Shallow hashing methods learn linear hash functions and rely on carefully-constructed discriminative features that are extracted from any hand-crafted feature extraction techniques or any representation-learning algorithms. On the other hand, the deep hashing methods combine the feature representation learning phase and the hashing phase into an end-to-end model and have demonstrated significant performance improvements over the hand-crafted feature-based hashing approaches [8,7,6,5,12,13,14,15,16].

Hashing methods can also be divided into unsupervised [2,17,18,8,19] and supervised hashing [20,21,22,23,4,24,25,26,27]. The works in [2,28] regress from the hash code of an image to its semantic label. Li et al. [14,15] predict the class label of an image given its hash code. On the other hand, the works in [3,8] preserve the consistency between the hash codes approximated from the similarity matrix and the hash codes approximated from the deep networks. A pairwise similarity objective or triplet ranking objective can also be formulated by randomly drawing the similar and dissimilar examples of an image from the dataset [7,6]. Our work, GenHash, also models the relationship between the hash code of an image and its semantic label. However, GenHash ensures that the hash codes of the synthetic samples are also consistent with their sampled labels. Furthermore, different from the previous triplet-ranking-based hashing methods, the contrastive samples (similar and dissimilar images) in our triplet-ranking objective (in Section 3) are synthetic (i.e., generated from a generative model) instead of being drawn from the same empirical datasets. The primary reason is to improve the generalization of the learned hash function. GenHash is also orthogonal to OrthoHash [29] and the works in [30,9,10], all of which focus on improving the quantization aspect of learning the hash function.

Generative Supervised Hashing: Supervised methods can easily overfit with limited labeled data. Some methods overcome such a limitation by synthesizing additional training data to improve the generalization of the hash functions [31,32]. These methods employ the popular Generative Adversarial Network (GAN) to synthesize the contrastive images. The use of generative models in hashing is currently limited to only data synthesis. Yet, generative models can benefit other downstream problems such as data imputation and out-of-distribution robustness. Our work belongs to the deep, supervised hashing category and we aim to jointly learn an energy-based generative model and the hash function in an end-to-end manner. Borrowing the strengths of EBM, we improve the retrieval performance over the GAN-based approaches and significantly improve the robustness of the hash function in terms of handling missing data and out-of-distribution retrieval.

2.2 Energy-based Generative Models

The works in [33,34] propose a powerful generative model, called generative cooperative network (CoopNets), which can generate realistic image and video patterns. The CoopNets framework jointly trains an energy-based model (i.e., descriptor network) and a latent variable model (i.e., generator network) via a cooperative learning scheme, where the bottom-up descriptor network is trained by MCMC-based maximum likelihood estimation [35], while the top-down generator learns from the descriptor and serves as a fast initializer for the MCMC of the descriptor. While the CoopNets framework avoids mode collapse and the bottom-up descriptor is a valid model for both representation and generation, it still employs two separate networks that must be carefully designed together to ensure the model converges to a good local minima [33]. This problem also exists in GANs.

Du et al. [36] propose a scalable single-network EBM for the image generation task. The EBM can generate a realistic image and exhibits attractive properties of EBM such as out-of-distribution and adversarial robustness. Grathwohl et al. [37] reinterpreted the discriminative classification task, which estimate the conditional probability $p(x|y)$, with an energy-based model for the joint probability $p(x, y)$. To estimate the intractable partition function, the authors use MCMC sampling through the Langevin dynamics. Specifically, they build upon the persistent contrastive divergence (PCD) [38] and maintain a replay buffer to propagate the MCMC chains during training. This allows shorter mixing times than initialization of the chains from random noise, while occasionally reinitializing samples from random noise in the buffer allows the training process to explore different modes of the model. Our paper studies generative hashing based on the framework of a single EBM with multipurpose objectives. However, as we shall see later, the current PCD training procedure of existing EBM works does not work well for contrastive hash function learning where the loss function involves data synthesis of similar and dissimilar examples. Instead, we propose to train the EBMs by mixing between an exploration buffer and an exploitation buffer.

3 Multipurpose Generative Hashing Network

The proposed Multipurpose Generative Hashing Network consists of a shared representation network and multiple lightweight heads. They are jointly trained by an MCMC-based learning algorithm, as described in Figure 1.

3.1 Problem Statement

Given a dataset $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ of n images, the goal of a hashing method is to learn a discrete-output, nonlinear mapping function $\mathcal{H} : x \rightarrow \{-1, 1\}^K$, which encodes each image x into a K-bit binary vector such that the similarity structure between the images is preserved in the discrete space. In the supervised hashing setting, each example $x_i \in \mathcal{X}$ is associated with a label c_i. Note that

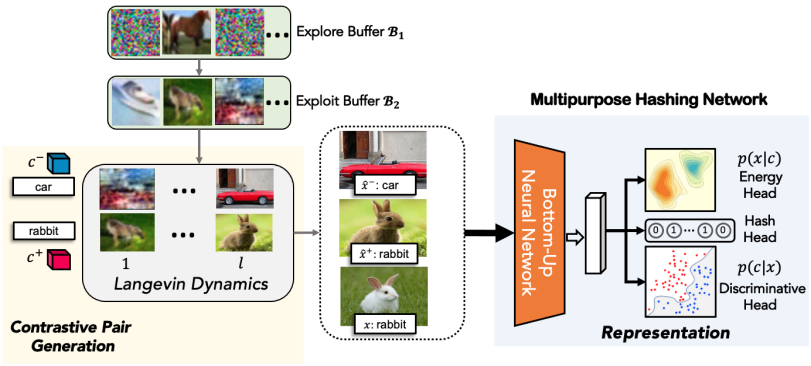

Fig. 1: GenHash is a multipurpose hashing network (light blue block) that describes the images in multiple ways, including an explicit density model $p(x|c)$, a discriminative model $p(c|x)$, and a hashing model, all of which share a base bottom-up representational network. The multipurpose hashing network is trained by a loss including negative maximum likelihood, triplet-ranking loss, and classification loss. To compute the triplet-ranking loss, GenHash relies on the Contrastive Pair Generation process (light yellow block) that takes a label c^+ as input and synthesizes (i.e., samples from the replay buffer $\mathcal{B}_2$) a contrastive image pair $\{\hat{x}^+, \hat{x}^-\}$ from the same class c^+ and a different class c^-.

this is a point-wise label of an image. Another common supervised scenario has the pairwise similarity label for each pair of images. However, for most image applications, pair-wise labeling is significantly labor-intensive because a dataset of n images requires n^2 pairwise labelings.

3.2 Multipurpose Energy-based Model

The multipurpose EBM aims at representing the images from different perspectives. We propose to parameterize this network by a multi-headed bottom-up neural network, where each branch accounts for one different representation of the image. The proposed network assembles three types of representational models of data in a single network in the sense that all models share a base network but have separate lightweight heads built on top of the base network for different representational purposes. Let $f_0(x; \theta_0)$ be the shared base network with parameters θ_0. Next, we will describe the purpose of each head in more detail.

Conditional Energy head: The energy head h_E along with the base network f_0 specifies an energy function $f_E(x, c; \Theta_E)$, where observed image-label pairs (which come from the real data distribution) are assigned lower energy values than unobserved ones. For notational simplicity, let the parameters be $\Theta_E = (\theta_0, \theta_E)$ and the energy function be $f_E(x, c; \Theta_E) = h_E(c, f_0(x, \theta_0); \theta_E)$. With the energy function f_E, the energy head explicitly defines a probability distribution

of x given its label c in the form of an energy-based model as follows:

$$p(x|c;\Theta_E) = \frac{p(x,c;\Theta_E)}{\int p(x,c;\Theta_E)dx} = \frac{\exp[-f_E(x,c;\Theta_E)]}{Z(c;\Theta_E)}, \tag{1}$$

where $Z(c;\Theta_E) = \int \exp[-f_E(x,c;\Theta_E)]dx$ is the intractable normalizing constant. Eq. (1) is also called generative modeling of neural network f_E [35]. Specifically, fixing the label c, $f_E(x,c;\Theta_E)$ defines the value of the compatible solution x and $-f_E(x,c;\Theta_E)$ defines the conditional energy function. Note that, for each value c, there are many compatible solutions x, i.e., there are several x's with similar, low conditional energies.

The training of θ_E in this context can be achieved by maximum likelihood estimation, which will lead to the "analysis by synthesis" algorithm [39]. Given a set of training images with labels $\{(c_i, x_i)\}_{i=1}^n$, we train Θ_E by minimizing the negative log-likelihood (NLL):

$$\mathcal{L}_{\mathrm{E}}(\Theta_E) = -\frac{1}{n}\sum_{i=1}^{n} \log p(x_i|c_i;\Theta_E), \tag{2}$$

The gradient of the above loss function is given by

$$\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbb{E}_{p(x|c_i;\Theta_E)}\left[\frac{\partial f_E(x,c_i;\Theta_E)}{\partial \Theta_E}\right] - \frac{\partial f_E(x_i,c_i;\Theta_E)}{\partial \Theta_E}\right\}, \tag{3}$$

where the $\mathbb{E}_{p(x|c_i;\Theta_E)}$ denotes the intractable expectation with respect to $p(x|c_i;\Theta_E)$.

Following the works in [36,37], we use persistent contrastive divergence (PCD) [38] to estimate the intractable expectation since it only requires short-run MCMC chains. This gives an order of magnitude savings in computation compared to initializing new chains with a long mixing time at each iteration. Intuitively, the PCD supplies the learning process with initial solutions from a replay buffer of past generated samples. The learning process then refines these solutions at high-value region around a mode of the objective function. The model's parameters are then updated so that the objective function shifts its high-value region around the mode towards the observed solution. In the next iteration, the refined solution will (hopefully) get closer to the observed solution.

The MCMC sampling strategy with the replay buffer can be summarized in two steps: (i) the algorithm first samples $\hat{x}$ from a replay buffer $\mathcal{B}_1$ with a probability $p_{\mathcal{B}_1}$ and from uniform noise with a probability $1 - p_{\mathcal{B}_1}$, and then (ii) it refines $\hat{x}$ by finite steps of Langevin updates [40], which is an example of MCMC, to obtain final $\tilde{x}$, as follows:

$$\tilde{x}_{t+1} = \tilde{x}_t - \frac{\delta^2}{2}\frac{\partial f(\tilde{x}_t,c;\Theta_E)}{\partial \tilde{x}} + \delta\mathcal{N}(0,I_D), \tilde{x}_0 = \hat{x}, \tag{4}$$

where t indexes the Langevin time steps, and δ is the step size. The Langevin dynamics in Eq. (4) is a gradient-based MCMC, which is equivalent to a stochastic gradient descent algorithm that seeks to find the minimum of the objective

function defined by $f_E(x, c; \Theta_E)$. The replay buffer $\mathcal{B}_1$ stores past generated samples. Occasionally re-sampling from random uniform noise is crucial to the learning process since different modes of the model can be explored in training. On the other hand, between the parameters' update steps, the model only slightly changes, thus sampling from past samples, which should be reasonably close to the model distribution, allows the algorithm to simulate longer MCMC chains on the samples. However, we call $\mathcal{B}_1$ an explore buffer because of its primary function, which is to seek and cover possible modes of the model.

With the MCMC examples, we can compute the gradient of the negative log-likelihood objective by

$$\nabla(\Theta_E) \approx \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial f_E(\tilde{x}_i, c_i; \Theta_E)}{\partial \Theta_E} - \frac{\partial f_E(x_i, c_i; \Theta_E)}{\partial \Theta_E} \right]. \tag{5}$$

Contrastive Hashing head: The head h_H learns to represent the input images as binary codes. The hash head h_H and the base network f_0 form a hash function $f_H(x; \Theta_H) = h_H(f_0(x; \theta_0); \theta_H)$, where $\Theta_H = (\theta_0, \theta_H)$. The hash function aims at mapping images with similar high-level concepts to similar hash codes and those of unrelated concepts to dissimilar codes. With a generative model, one effective way to learn such hash function is: for each image x, we "draw" a positive sample x^+ that is conceptually similar to x and a negative sample x^- that is conceptually dissimilar to x, and train the hash function to produce similar hash codes for x and x^+, and dissimilar hash codes for x and x^-.

Such contrastive learning can be achieved by recruiting labeled generated samples from the inference of the conditional EBM. These synthetic samples from each class can be similarly generated using MCMC sampling. To avoid a long mixing time where the MCMC chains are initialized from random noise, we can reuse past generated samples from the replay buffer $\mathcal{B}_1$. However, $\mathcal{B}_1$ may contain several past samples that have been less rigorously refined through the Langevin dynamics (i.e., only refined a few times). These "young" samples can still be closer to random noise. Therefore, when being selected for contrastive hash learning, there is not a useful difference between samples from a different class. That is, the current sample x and their similar and dissimilar synthetic samples x^+ and x^-, respectively, do not form an informative contrastive triplet. Empirically, we observe that only relying on $\mathcal{B}_1$ for contrastive samples learn a hash function whose performance is significantly worse than desired.

This problem is further illustrated in Figure 2, where some samples from each class are similar to random noise. There are more of these samples in CIFAR10 than in MNIST because the generation of natural images in CIFAR10 requires a significantly more complex model than the generative model of MNIST. Even further into training, we observe that there are still similar MCMC samples due to the occasional sampling from random noise that requires longer MCMC chains when the data distribution is complex (as in the case of CIFAR10). One naive solution is to increase the number of Langevin steps; however, larger chains make the EBM significantly more computationally expensive to train.

(a) MNIST (b) CIFAR10

Fig. 2: Samples from the explore buffer $\mathcal{B}_1$. Images on each row are from the same semantic class. Some CIFAR10 images in this buffer are far from real images because $\mathcal{B}_1$ is a mixture of (i) *newly-initialized or younger samples* from the MCMC chain and (ii) *those samples that have been revised several times.*

We propose a simple, yet effective sampling strategy to solve this problem. First, we introduce a second replay buffer $\mathcal{B}_2$, where a sample $x \in \mathcal{B}_2$ is required to be initialized from supposedly longer MCMC chains. To avoid explicitly increasing the number of Langevin steps to achieve such longer MCMC chains, we leverage the existing replay buffer $\mathcal{B}_1$ by "copying" samples that have been revised several times in $\mathcal{B}_1$. Intuitively, while sampling from $\mathcal{B}_1$ allows the training process to explore different modes of the models, sampling from $\mathcal{B}_2$ exploits the learned modes. However, since we regularly copy the samples from $\mathcal{B}_1$ to $\mathcal{B}_2$ when they are "matured", sampling from $\mathcal{B}_2$ also explores all previously learned modes in for the contrastive hash learning.

Under this sampling strategy, to learn the contrastive hash function, for each observed image x and its label c, we sample a synthetic image x^+, conditioned on the label c, and a synthetic image x^-, conditioned on a different label $c^- \neq c$ using the replay buffer $\mathcal{B}_2$. The three examples form a real-synthetic triplet (x, x^+, x^-). The hash function f_H can be trained to minimize the Hamming distance (a discrete distance function that is typically approximated by the continuous L_2 distance) between $f_H(x)$ and $f_H(x^+)$ and maximize the distance between $f_H(x)$ and $f_H(x^-)$. This triplet-ranking loss is defined as follows:

$$\mathcal{L}_{\mathrm{H}}(\Theta_H) = ||f_H(x) - f_H(x^+)||_H + max(m - ||f_H(x) - h(x^-)||_H, 0) \quad (6)$$
$$s.t. \;\; f_H(x) \in \{-1, 1\}, f_H(x^+) \in \{-1, 1\}, f_H(x^-) \in \{-1, 1\}$$

where $||.||_H$ denotes the Hamming distance. The first term preserves the similarity between images with similar semantic concepts, while the second term penalizes the mapping of semantically dissimilar images to similar hash codes if their distance is within a margin m. Essentially, this is a contrastive objec-

tive that avoids collapsed solutions because it only considers the dissimilar pairs having distances within a certain margin to contribute to the loss.

The objective of $\mathcal{L}_h$ is a discrete optimization problem, hence it is computationally intractable to solve and is not suitable for a gradient-based backpropagation algorithm. A natural solution is to approximate the discrete constraints with real-valued output and replace the Hamming distance with Euclidean distance. For the thresholding procedure, a commonly-used trick is to employ the *tanh* or *sigmoid* function. However, we find that *tanh* or *sigmoid* makes the learning process more difficult to converge to good local optima. To overcome this, we propose to directly regularize the real-valued output of the hash function to the desired discrete values. The final triplet-ranking loss is as follows:

$$\begin{aligned}\mathcal{L}_{\mathrm{H}}(\Theta_H) = &||f_H(x) - f_H(x^+)||_2 + \max(m - ||f_H(x) - f_H(x^-)||_2, 0) \\ &+ \lambda(|||f_H(x)| - 1||_2 \quad + \quad |||f_H(x^+)| - 1||_2 + |||f_H(x^-)| - 1||_2)\end{aligned} \tag{7}$$

where $f_H(.)$ is now the relaxed function with real-valued vector outputs and $|.|$ is element-wise absolute operation. The first and second terms in the objective function approximate the Hamming distances in Eq. (7). The last term minimizes the quantization error of approximating the discrete solution with the real-value relaxation. Intuitively, it centers the relaxed, continuous output of the hash function around the desired binary value of -1 or 1. Note that the *max* operation is non-differentiable; however, we can define the subgradient of the *max* function to be 1 at the non-differential points.

Discriminative head: Image labels provide not only knowledge for training a classification model but also a supervised signal for extracting high-level information of the images. On the other hand, the learned hash codes should also capture high-level abstractions of the images, therefore should be predictive of the image labels. This relationship can be modeled through a multi-class classification problem. Specifically, we propose a classification head h_C that predicts the class label of an image given its hash code. For each image x_i, let $\hat{c}_i$ be the predicted label. The multi-class classification loss can be defined as follows:

$$\mathcal{L}_C(\Theta_C) = -\frac{1}{n}\sum_{i=1}^{n} c_i \log \frac{e^{\theta_{c_i,:}^T h(x)}}{\sum_j e^{\theta_{j,:}^T h(x_i)}} \tag{8}$$

where $\theta_C \in \mathbb{R}^{K \times L}$ is the parameter of the linear layer that maps each hash code into the class labels. We additionally denote $\Theta_C = (\theta_0, \theta_C)$ as the parameters of this classification network. This objective function is optimal when the discrete space is approximately linear separable with respect to the class labels. In other words, the hash codes of the images from the same semantic class have small Hamming distances between them, while the hash codes of the images from different classes have larger Hamming distances.

3.3 Optimization

At each iteration, the energy-based model $p(x|c;\Theta_E)$ samples synthetic contrastive image pairs by following the strategy described in the previous section.

With synthetic images, we train the multipurpose hashing network to simultaneously describe the images from multiple representational perspectives. The overall training objective of the network, which combines the negative log-likelihood $\mathcal{L}_{\mathrm{E}}(\Theta_E)$, the triplet-ranking loss $\mathcal{L}_{\mathrm{H}}(\Theta_H)$, and the classification loss $\mathcal{L}_{\mathrm{C}}(\Theta_C)$, is given by:

$$\mathcal{L}(\Theta_E, \Theta_H, \Theta_C) = \mathcal{L}_E(\Theta_E) + \beta_H \mathcal{L}_H(\Theta_H) + \beta_C \mathcal{L}_C(\Theta_C) \tag{9}$$

where β_H and β_C are parameters to balance the weight between different losses.

In general, EBMs are less computationally efficient than GAN-based models [36]. However, the increased computational cost (compared to GAN-based methods) is only in the training phase, which can be mitigated with larger hardware and distributed training. With short-run MCMCs (typically only 15-20 Langevin steps in our experiments), we can already significantly reduce the training time of the EBM component. During the testing phase, since we only use the hash function and discard the remaining components (Figure 1b), the computation is similar to the computation of the models in other hashing methods.

4 Experiments

In this section, we present the evaluation results on several real-world datasets to demonstrate the effectiveness of the proposed method.

4.1 Experimental Setup

Datasets: We evaluate our method on three widely-used datasets in the image hashing domain: **NUS-WIDE**, **COCO** and **CIFAR-10**.
Evaluation Metrics: We evaluate the retrieval performance of the methods using the standard retrieval metrics in image hashing: Mean Average Precision at K (mAP@K) and Precision at K (P@K).
Baselines: We compare our method against several representative approaches from image hashing. ITQ [4], BRE [3], and KSH [41]) are shallow supervised hashing approaches. CNNH [8], SDH [2], DNNH [7], FastHash [28], DHN [6], DSDH [14], DVStH [42], HashNet [5], CSQ [43], and the state-of-the-art generative method, HashGAN [19] are deep supervised hashing approaches. Additionally, we include the results of HashGAN-1, which is the HashGAN method where the GAN model and the hash function are jointly learned in one stage.

The complete experimental setup, including the dataset details, evaluation metric calculations, and implementation details, are provided in the Supplement.

4.2 Retrieval Results

In this section, we present the results of querying for similar images. Table 1 shows the mAP results for all the methods. Compared to the shallow hashing methods, GenHash improves at least 14% on NUS-WIDE, 13% on CIFAR-10,

Table 1: Mean Average Precision (mAP) for different number of bits.

Method	NUS-WIDE			CIFAR-10			COCO		
	16 bits	32 bits	48 bits	16 bits	32 bits	48 bits	16 bits	32 bits	48 bits
ITQ [4]	0.460	0.405	0.373	0.354	0.414	0.449	0.566	0.562	0.530
BRE [3]	0.503	0.529	0.548	0.370	0.438	0.468	0.592	0.622	0.630
KSH [41]	0.551	0.582	0.612	0.524	0.558	0.567	0.521	0.534	0.534
SDH [2]	0.588	0.611	0.638	0.461	0.520	0.553	0.555	0.564	0.572
CNNH [8]	0.570	0.583	0.593	0.476	0.472	0.489	0.564	0.574	0.571
DNNH [7]	0.598	0.616	0.635	0.559	0.558	0.581	0.593	0.603	0.605
FastHash [28]	0.502	0.515	0.516	0.524	0.566	0.597	0.601	0.609	0.612
DHN [6]	0.637	0.664	0.669	0.568	0.603	0.621	0.677	0.701	0.695
DSDH [14]	0.650	0.701	0.705	0.655	0.660	0.682	0.659	0.688	0.710
DVStH [42]	0.661	0.680	0.698	0.667	0.695	0.708	0.689	0.709	0.713
HashNet [5]	0.662	0.699	0.711	0.643	0.667	0.675	0.687	0.718	0.730
CSQ [43]	0.701	0.713	0.720	0.646	0.699	0.709	0.679	0.699	0.714
HashGAN [19]	0.715	0.737	0.744	0.668	0.731	0.735	0.697	0.725	0.741
GenHash	**0.742**	**0.754**	**0.773**	**0.711**	**0.739**	**0.778**	**0.747**	**0.768**	**0.775**

Table 2: Precision@1000 for different number of bits.

Method	NUS-WIDE			CIFAR-10			COCO		
	16 bits	32 bits	48 bits	16 bits	32 bits	48 bits	16 bits	32 bits	48 bits
ITQ [4]	0.489	0.572	0.590	0.289	0.271	0.305	0.489	0.518	0.545
BRE [3]	0.521	0.603	0.627	0.398	0.445	0.471	0.520	0.535	0.559
KSH [41]	0.598	0.656	0.667	0.580	0.612	0.641	0.519	0.540	0.558
SDH [2]	0.640	0.702	0.712	0.655	0.671	0.651	0.696	0.695	0.710
CNNH [8]	0.601	0.651	0.672	0.533	0.545	0.578	0.671	0.690	0.718
DNNH [7]	0.620	0.689	0.707	0.651	0.678	0.691	0.713	0.701	0.728
DHN [6]	0.655	0.713	0.726	0.659	0.701	0.725	0.703	0.731	0.750
DSDH [14]	0.658	0.728	0.752	0.678	0.710	0.729	0.721	0.735	0.754
HashNet [5]	0.680	0.729	0.741	0.720	0.721	0.741	0.745	0.746	0.753
CSQ [43]	0.701	0.741	0.750	0.725	0.735	0.741	0.749	0.742	0.749
HashGAN [19]	0.720	0.759	0.772	0.735	0.751	0.762	0.755	0.768	0.783
GenHash	**0.749**	**0.780**	**0.808**	**0.780**	**0.799**	**0.823**	**0.789**	**0.795**	**0.811**

and 20% on COCO. Compared to the state-of-the-art deep hashing method that does not have data synthesis (i.e., HashNet), GenHash improves at least 9%, 9%, and 5% on NUS-WIDE, CIFAR-10, and COCO, respectively. GenHash outperforms HashGAN, the state-of-the-art supervised, data-synthesis method by statistically significant margins in all the datasets.

The mAP results provide empirical evidence to support our discussion in Sections 1 and 2. **First**, generating synthetic data improves the performance of a supervised hashing method. This could be explained by the fact that generative models improve the amount of "labeled" training data and increase its diversity, both of which improve the method's generalization capacity. **Second**, we can observe that the performance of HashGAN significantly decreases without the fine-tuning step (HashGAN-1's results). Training GAN-based models is difficult with problems such as mode collapse; thus in GAN-based models such as HashGAN, a second fine-tuning step, where only the hash function is trained with the synthetic data while the other components are fixed, is needed to avoid difficul-

ties in simultaneously training the generator and discriminator. This increases the computational requirement of the GAN-based methods. Finally, GenHash, which simultaneously trains the generative model and the hash function, has better retrieval performance than the state-of-the-art, two-stage HashGAN. This supports our claim that the one-stage scheme learns better similarity-preserving hash codes of the images.

In addition, we present the Precision@1000 results in Table 2. The Precision@1000 is calculated at the common retrieval threshold (1000) in image applications. Similarly, the proposed GenHash significantly outperforms all the compared methods.

4.3 Out-of-distribution Retrieval

In this section, we show that GenHash, which is a multipurpose EBM, exhibits better out-of-distribution (OOD) robustness in retrieval than other methods. In real-world retrieval applications, the arrival of new data instances or new data format are common. This results in conceptual drift or change in the underlying data distribution where the hashing methods are trained on. A hashing method that is robust (i.e., its performance is not significantly worse) to slight changes in such underlying distributional change in the data is preferred because it takes a longer time for the trained model to become obsolete.

Table 3: OOD Retrieval.

Train/Test	HashNet	CSQ	HashGAN	GenHash
SVHN/*MNIST*	0.181	<u>0.517</u>	0.354	**0.609**
SVHN/*SVHN*	0.837	0.854	<u>0.889</u>	**0.895**
MNIST/*SVHN*	0.193	0.273	<u>0.280</u>	**0.498**
MNIST/*MNIST*	0.957	<u>0.991</u>	0.990	**0.991**

Table 4: Data-corruption Retrieval.

	Type	HashNet	HashGAN	GenHash
SnP	Clean	0.513	0.608	0.680
	Corrupted	0.223	0.281	0.652
RRM	Clean	0.471	0.615	0.654
	Corrupted	0.243	0.298	0.607

We propose to simulate a minor but realistic distributional change in the data as follows. In the learning phase, each hashing method is trained on a source dataset. In the testing or evaluation phase, we use a different test dataset that is conceptually similar to the source dataset but comes from a (slightly) different data distribution. We choose MNIST and SVHN as conceptually-related datasets. One dataset is selected as both the train and retrieval sets, while the test queries are sampled from the other dataset.

Table 3 shows the retrieval results of GenHash, HashNet (a deep hashing method), HashNet and HashGAN (a GAN-based hashing method). As can be observed, GenHash significantly outperforms both HashNet and HashGAN in OOD retrieval, with more than 25% when using MNIST for querying, and 20% when using SVHN for querying. GenHash's mAP performance is still roughly more than 50% when the test data distribution changes. This makes GenHash still useful in practice, while in other methods, the retrieval performance significantly drops closer to the performance of a random retrieval.

While the OOD retrieval performance falls significantly, compared to the retrieval performance using data from the same distribution as that of the training data, data-synthesis methods (HashGAN and GenHash) are more robust toward distributional changes, compared to the conventional deep hashing methods

HashNet and CSQ. In addition, when being trained on a more complex dataset (SVHN), the retrieval performance of GenHash significantly improves in our OOD tests, while the OOD retrieval performances of the other methods only slightly improve.

4.4 Missing-data Robustness in Retrieval

GenHash is a multipurpose EBM. Similar to other explicit generative EBMs [37,36], we can additionally model the energy function of x. This provides us with an important advantage over other generative models: we can revise (or reconstruct) a sample with corruption by initializing the input chain into the Langevin dynamics with the corrupted samples. Through the Langevin revision, the corrupted samples can be re-constructed. Note that, this feature of the EBM is not immediately available in other generative hashing methods, such as HashGAN.

We perform the missing data experiments as follows. First, we assume that both training data and test data may contain corrupted input images. During training of GenHash, we train on the clean input as mentioned previously. For corrupted input we initialize the MCMC chains with the corrupted samples and revise these samples through the Langevin dynamics. We corrupt the images in both the training and test sets of the CIFAR10 dataset. We corrupt 20% of the data using salt-and-pepper noise (denoted by SnP) or random rectangular mask (denoted by RRM, where the rectangles randomly cover approximately 10-20% of the images at random locations) on the images for both training and query sets. Then, the model in each hashing method is trained with the corrupted training set and the evaluation is performed on the corrupted test set.

In Table 4, we show the results for the missing-data robustness experiments. As can be observed, the performances of the baseline methods, including the generative hashing HashGAN method, are significantly degraded when there are corruptions in the data. On the other hand, GenHash's retrieval performance only slightly drops when the data is corrupted. This shows the advantages of the proposed EBM-based multipurpose generative hashing network.

5 Conclusion

This paper proposes a unified generative framework, called GenHash to solve the image hashing problem. The framework learns a multipurpose hashing network to represent images from multiple perspectives, including classification, hashing, and probability density estimation. This approach learns high-quality binary hash codes and achieves state-of-the-art retrieval performance on several benchmark datasets. Furthermore, GenHash is significantly more robust to out-of-distribution retrieval compared to the existing methods and can handle significant corruption in the data with trivial drops in the retrieval performance. In GenHash, we also train a single EBM-based network, which makes it easier for the practitioner to design better architectures. This is preferred compared to other GAN-based approaches because designing the discriminator and generator networks is not a trivial task with various problems when one network has more capacity than the other.

References

1. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge university press (2014)
2. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 37–45
3. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in neural information processing systems. (2009) 1042–1050
4. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE transactions on pattern analysis and machine intelligence **35** (2012) 2916–2929
5. Cao, Z., Long, M., Wang, J., Yu, P.S.: Hashnet: Deep learning to hash by continuation. In: Proceedings of the IEEE international conference on computer vision. (2017) 5608–5617
6. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: Thirtieth AAAI Conference on Artificial Intelligence. (2016)
7. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3270–3278
8. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI'14, AAAI Press (2014) 2156–2162
9. Doan, K.D., Reddy, C.K.: Efficient implicit unsupervised text hashing using adversarial autoencoder. In: Proceedings of The Web Conference 2020. WWW '20, New York, NY, USA, Association for Computing Machinery (2020) 684–694
10. Doan, K.D., Manchanda, S., Badirli, S., Reddy, C.K.: Image hashing by minimizing discrete component-wise wasserstein distance. arXiv preprint arXiv:2003.00134 (2020)
11. Doan, K.D., Yadav, P., Reddy, C.K.: Adversarial factorization autoencoder for look-alike modeling. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), Beijing, China (2019) 2803–2812
12. Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 2475–2483
13. Cao, Y., Long, M., Wang, J., Zhu, H., Wen, Q.: Deep quantization network for efficient image retrieval. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 30. (2016)
14. Li, Q., Sun, Z., He, R., Tan, T.: Deep supervised discrete hashing. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. (2017) 2479–2488
15. Gui, J., Li, P.: R 2 sdh: Robust rotated supervised discrete hashing. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. (2018) 1485–1493
16. Jiang, Q.Y., Li, W.J.: Asymmetric deep supervised hashing. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 32. (2018)
17. Yang, H.F., Lin, K., Chen, C.S.: Supervised learning of semantics-preserving hash via deep convolutional neural networks. IEEE transactions on pattern analysis and machine intelligence **40** (2018) 437–451

18. Ge, T., He, K., Sun, J.: Graph cuts for supervised binary coding. In: European Conference on Computer Vision, Springer (2014) 250–264
19. Cao, Y., Liu, B., Long, M., Wang, J.: Hashgan: Deep learning to hash with pair conditional wasserstein gan. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 1287–1296
20. Huang, S., Xiong, Y., Zhang, Y., Wang, J.: Unsupervised triplet hashing for fast image retrieval. In: Proceedings of the on Thematic Workshops of ACM Multimedia 2017, ACM (2017) 84–92
21. Lin, K., Lu, J., Chen, C.S., Zhou, J.: Learning compact binary descriptors with unsupervised deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 1183–1192
22. Huang, C., Change Loy, C., Tang, X.: Unsupervised learning of discriminative attributes and visual representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 5175–5184
23. Do, T.T., Doan, A.D., Cheung, N.M.: Learning to hash with binary deep neural network. In: European Conference on Computer Vision, Springer (2016) 219–234
24. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in neural information processing systems. (2009) 1753–1760
25. Salakhutdinov, R., Hinton, G.: Semantic hashing. International Journal of Approximate Reasoning **50** (2009) 969–978
26. Yang, E., Liu, T., Deng, C., Liu, W., Tao, D.: Distillhash: Unsupervised deep hashing by distilling data pairs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2946–2955
27. Dizaji, K.G., Zheng, F., Nourabadi, N.S., Yang, Y., Deng, C., Huang, H.: Unsupervised deep generative adversarial hashing network. In: CVPR 2018. (2018)
28. Gui, J., Liu, T., Sun, Z., Tao, D., Tan, T.: Fast supervised discrete hashing. IEEE transactions on pattern analysis and machine intelligence **40** (2017) 490–496
29. Hoe, J.T., Ng, K.W., Zhang, T., Chan, C.S., Song, Y.Z., Xiang, T.: One loss for all: Deep hashing with a single cosine similarity based learning objective. In: Advances in Neural Information Processing Systems. (2021)
30. Doan, K.D., Yang, P., Li, P.: One loss for quantization: Deep hashing with discrete wasserstein distributional matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2022) 9447–9457
31. Qiu, Z., Pan, Y., Yao, T., Mei, T.: Deep semantic hashing with generative adversarial networks. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. (2017) 225–234
32. Gao, R., Lu, Y., Zhou, J., Zhu, S.C., Nian Wu, Y.: Learning generative convnets via multi-grid modeling and sampling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 9155–9164
33. Xie, J., Lu, Y., Gao, R., Zhu, S.C., Wu, Y.N.: Cooperative training of descriptor and generator networks. IEEE transactions on pattern analysis and machine intelligence **42** (2018) 27–45
34. Xie, J., Zheng, Z., Fang, X., Zhu, S.C., Wu, Y.N.: Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2021)
35. Xie, J., Lu, Y., Zhu, S.C., Wu, Y.: A theory of generative convnet. In: International Conference on Machine Learning. (2016) 2635–2644
36. lun Du, Y., Mordatch, I.: Implicit generation and modeling with energy based models. In: NeurIPS. (2019)

37. Grathwohl, W., Wang, K., Jacobsen, J., Duvenaud, D., Norouzi, M., Swersky, K.: Your classifier is secretly an energy based model and you should treat it like one. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net (2020)
38. Tieleman, T.: Training restricted boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on Machine learning. (2008) 1064–1071
39. Grenander, U., Miller, M.I., Miller, M., et al.: Pattern theory: from representation to inference. Oxford university press (2007)
40. Zhu, S.C., Mumford, D.: Grade: Gibbs reaction and diffusion equations. In: International Conference on Computer Vision (ICCV). (1998) 847–854
41. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (2012) 2074–2081
42. Liong, V.E., Lu, J., Duan, L.Y., Tan, Y.P.: Deep variational and structural hashing. IEEE transactions on pattern analysis and machine intelligence **42** (2018) 580–595
43. Yuan, L., Wang, T., Zhang, X., Tay, F.E., Jie, Z., Liu, W., Feng, J.: Central similarity quantization for efficient image and video retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 3083–3092