Pairwise Supervised Hashing with Bernoulli Variational Auto-Encoder and Self-Control Gradient Estimator

Siamak Zamani Dadaneh[†⋆], **Shahin Boluki**[†⋆], **Mingzhang Yin**[‡], **Mingyuan Zhou**[‡], **Xiaoning Qian**[†]
[†] Texas A&M University [‡]The University of Texas at Austin
[⋆]Equal contribution

Abstract

Semantic hashing has become a crucial component of fast similarity search in many large-scale information retrieval systems, in particular, for text data. Variational autoencoders (VAEs) with binary latent variables as hashing codes provide state-of-the-art performance in terms of precision for document retrieval. We propose a pairwise loss function with discrete latent VAE to reward within-class similarity and between-class dissimilarity for supervised hashing. Instead of solving the optimization relying on existing biased gradient estimators, an unbiased low-variance gradient estimator is adopted to optimize the hashing function by evaluating the non-differentiable loss function over two correlated sets of binary hashing codes to control the variance of gradient estimates. This new semantic hashing framework achieves superior performance compared to the state-of-the-arts, as demonstrated by our comprehensive experiments.

1 INTRODUCTION

The problem of *similarity search* is to find the most similar items in a large collection to a query item of interest (Andoni, 2009). Fast similarity search is at the core of many information retrieval applications, such as collaborative filtering (Sarwar et al., 2001), content-based retrieval (Lew et al., 2006), and caching (Pandey et al., 2009). In particular, with the explosion of information on Internet in the form of text data, searching for relevant content in such gigantic databases is critical.

Traditional text similarity search methods are conducted in the space of original word counts, and thus can be computationally prohibitive due to high dimensions. Therefore, many research efforts have been devoted to employ approximate similarity search approaches in lower embedding dimensions. Semantic hashing (Salakhutdinov and Hinton, 2009) is an effective way of accelerating similarity search by designing compact binary codes in a low-dimensional space so that semantically similar documents are mapped to similar codes. The similarity between documents is evaluated by simply computing the pairwise Hamming distances between the hashing codes, i.e., the number of bits that are different between two codes. Furthermore, exploiting binary hashing codes is much more memory efficient, especially for big text corpora.

Deep learning has dramatically improved the state-of-the-arts in many applications, including speech recognition, computer vision, and natural language processing (LeCun et al., 2015). Learning expressive feature representations for complex data lies at the core of deep learning. Recently, deep generative models such as variational autoencoder (VAE) have been proposed for neural semantic hashing (Chaidaroon and Fang, 2017). Employing VAEs for document hashing has two major benefits. First, they can learn flexible nonlinear distributed representations of the original high-dimensional documents. Second, due to amortized computational cost for inference in VAEs, the hashing codes for new documents can be simply calculated with one pass through the encoder network.

In their basic form, VAEs assume that latent variables are distributed according to a multivariate normal distribution. The continuous latent representations are then binarized to obtain the hashing codes corresponding to the documents. As a result, the information contained in the continuous representations may be lost during the binarization step. Shen et al. (2018) have developed a VAE framework with Bernoulli latent variables as hashing codes, obviating the need for the binarization step. To optimize the VAE model parameters, straight-through (ST) gradient estimator (Bengio et al., 2013) with respect to binary latent variables is adopted in Shen et al. (2018). While easy to implement, ST gradient estimator is clearly

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

biased, and hence it can undermine the performance of the VAE with binary latent representations as hashing codes to capture the semantic similarities of documents.

In this paper, we aim to develop a faithful discrete VAE with Bernoulli latent variables as binary hashing codes that can be inferred without bias. When additional information such as document labels can be leveraged for a more targeted similarity search, we propose a pairwise supervised hashing (PSH) framework to derive better hashing codes, with two main objectives: (1) to learn informative binary codes, capable of reconstructing the original word counts; (2) to minimize the distance between the hashing codes of documents from the same class and maximize this distance for documents from different classes. The first objective can be achieved through maximizing the evidence lower bound (ELBO) with weighted Kullback–Leibler (KL) regularization (Alemi et al., 2018; Zhao et al., 2017; Higgins et al., 2017). To achieve the second objective, we add a pairwise loss function to reward within-class similarity and between-class dissimilarity. This end-to-end generative framework is distinct from previous methods training a neural network classifier with latent variables as inputs and document labels as outputs for supervised hashing (Shen et al., 2018; Chaidaroon and Fang, 2017), which fail to extract useful similarity patterns for efficient search as they consider documents in isolation.

We exploit stochastic gradient based optimization to learn this Bernoulli VAE hashing model. The main difficulty arises due to the binary hashing code based latent representations. The recently proposed augment-REINFORCE-merge (ARM) (Yin and Zhou, 2019; Yin et al., 2020) gradient estimator provides a natural solution with unbiased low-variance gradient updates during the training of our discrete VAE. With a single Monte Carlo sample, the estimated gradient is the product of uniform random noise and the difference of the objective functions with two vectors of correlated binary latent variables as inputs. Applying the ARM gradient leads to not only fast convergence, but also low negative evidence lower bounds for variational inference, thus increasing the ability to reconstruct the original word counts from the binary hashing codes.

Comprehensive experiments conducted on benchmark datasets for both supervised and unsupervised hashing demonstrate the superior performance of our proposed framework in terms of precision for document retrieval. In particular, PSH gains significantly better performance for short hashing codes making it more attractive for practical applications with limited memory budget.

Our main contributions to hashing-based similarity retrieval include:

- We propose a flexible discrete VAE-based framework, directly with binary hashing codes as latent representations, for both unsupervised and supervised semantic hashing. With unbiased and low-variance ARM gradient estimator, efficient variational inference as well as one-pass hashing code generation given new documents can be achieved without commonly adopted continuous relaxation.
- A novel pairwise loss function is defined for supervised hashing, obviating the need for access to ordinal labels in the training phase. ARM gradient estimator is specially useful for learning when the loss function involves non-differentiable components with binary random variables for hashing codes.
- Our method is highly scalable, applicable to large-scale data. Our comprehensive experimental results with ablation studies have verified the advantage of our direct hashing code based VAE with ARM variational inference, as well as the benefits from our new loss function with the expected pairwise loss. More importantly, our new method consistently outperforms state-of-the-art methods over several widely used benchmark datasets.

The remainder of this paper is organized as follows. In Section 2, we present the main methodology, including the structure of Bernoulli VAE for document hashing, optimization using ARM gradient estimator, and pairwise hashing in the supervised scenario. Section 3 discusses related work. Section 4 provides comprehensive experimental results in supervised as well as unsupervised settings, with comparison with existing hashing methods. Section 5 concludes the paper.

2 METHODS

2.1 Hashing Using Bernoulli VAEs

Let $\boldsymbol{x}$ and $\boldsymbol{z}$ denote the input document and its corresponding binary hashing code. Specifically, $\boldsymbol{x} \in \mathcal{Z}_{+}^{|V|}$ is a vector of word counts for the input document, where $|V|$ is the size of the vocabulary V. Under the variational auto-encoder (VAE) framework (Kingma and Welling, 2013; Rezende et al., 2014), a generative (decoding) model $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ reconstructs the input document from the binary hashing code, while an inference (encoder) model $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$ infers the code $\boldsymbol{z}$ from the input document $\boldsymbol{x}$. The model parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$ are the weights of neural networks employed by the decoder and encoder.

2.1.1 Decoder Structure

To build the decoder, we follow the same procedure as in Chaidaroon and Fang (2017); Shen et al. (2018), and utilize a *softmax* decoding function. Assuming that t_i, the ith token within document $\boldsymbol{x}$, is the wth word of the vocabulary, we denote its one-hot vector representation by $\boldsymbol{o}_w \in \{0,1\}^{|V|}$, a vector with a one at wth element and zeros elsewhere. The decoder network comprises a linear transformation of the latent binary hashing code $\boldsymbol{z}$, followed by a softmax function which outputs the likelihood of individual tokens as:

$$p_{\boldsymbol{\theta}}(t_i = w \,|\, \boldsymbol{z}) = \frac{\exp\left(\boldsymbol{z}^T E \boldsymbol{o}_w + b_w\right)}{\sum_{j=1}^{|V|} \exp\left(\boldsymbol{z}^T E \boldsymbol{o}_j + b_j\right)}, \quad \text{for } w \in V, \tag{1}$$

where $E \in \mathbb{R}^{K \times |V|}$ can be interpreted as a word embedding matrix and $\boldsymbol{b} = [b_1, ..., b_{|V|}]^T$ are the word biases. Thus, the decoder parameters to be learned are $\boldsymbol{\theta} = \{E, \boldsymbol{b}\}$. Given the individual token likelihoods in (1) and the word counts $\boldsymbol{x}$, the document likelihood can be computed as

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}) &= \sum_i \log p_{\boldsymbol{\theta}}(t_i \,|\, \boldsymbol{z}) \\ &= \sum_{w: x_w > 0} x_w \log p_{\boldsymbol{\theta}}(x_w \,|\, \boldsymbol{z}). \end{aligned} \tag{2}$$

To exploit the relevance of words in documents, we replace the log weights in (2) with Term Frequency Inverse Document Frequency (TF-IDF) (Ramos et al., 2003). Hence, we use the following modified reconstruction term in the optimization procedure of the ELBO explained in latter sections:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\Big[\sum_{w: x_w > 0} \text{TF-IDF}_w \times \log p_{\boldsymbol{\theta}}(x_w \,|\, \boldsymbol{z})\Big].$$

2.1.2 Encoder Structure

We employ the amortized inference of hashing codes for documents by constructing an inference network as $f_{\boldsymbol{\phi}}(\boldsymbol{x})$ to approximate the true posterior distribution $p(\boldsymbol{z} \,|\, \boldsymbol{x})$ by $q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})$. More precisely, the approximate posterior for the K-dimensional latent code $\boldsymbol{z} \in \{0,1\}^K$ is expressed as

$$q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \prod_{k=1}^{K} \text{Bernoulli}\big(z_k; \sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})_k)\big), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, and $f_{\boldsymbol{\phi}}(\boldsymbol{x})_k$ is the kth element of the encoder neural network's output. In the training phase, latent codes are sampled using the Bernoulli distributions in (3) and subsequently fed into the decoder network, while in the testing phase, hard thresholding the means at 0.5 is used to infer the hashing codes. Finally, we place independent Bernoulli priors on the components of latent codes as $p(\boldsymbol{z}) = \prod_{k=1}^{K} \text{Bernoulli}(z_k; \gamma_k)$, where $\gamma_k \in [0,1]$. Our Bernoulli distributed latent variables obviate the need for a separate binarization step; and hence they are more capable of capturing the semantic structure of input documents.

2.1.3 Variational Inference

To estimate the parameters of encoder and decoder networks, the VAE framework optimizes ELBO defined as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) &:= \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\Big[\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z})\big] \\ &\qquad -\text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) || p(\boldsymbol{z}))\Big] \\ &\leq \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big], \end{aligned} \tag{4}$$

where KL is the Kullback–Leibler divergence and $p_{\mathcal{D}}(\boldsymbol{x})$ is the empirical distribution of the inputs. Since both prior and approximate posterior are Bernoulli distributions, the KL term can be computed in the closed form:

$$\begin{aligned} \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) || p(\boldsymbol{z})) &= \sum_k \Big\{\sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})_k) \log \frac{\sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})_k)}{\gamma_k} \\ &+ (1 - \sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})_k)) \log \frac{1 - \sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})_k)}{1 - \gamma_k}\Big\}. \end{aligned} \tag{5}$$

In practice, to extract useful latent representations and to avoid latent variable collapse (Dieng et al., 2019), a modification of ELBO with the weighted KL term is employed:

$$\begin{aligned} \mathcal{L}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi}) &:= \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\Big[\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z})\big] \\ &\qquad -\lambda \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) || p(\boldsymbol{z}))\Big], \end{aligned}$$

where $0 < \lambda < 1$. The parameters are then estimated by stochastic gradient optimization of $\mathcal{L}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi})$. In what follows, we drop the expectation with respect to the empirical distribution to simplify the notations.

2.2 Pairwise Supervised Hashing (PSH)

When training data come with side information such as document labels, the previously discussed discrete VAE is not ready to take advantage of that. To mitigate such a shortcoming for deriving better latent hashing codes in this generative framework, we add a supervised layer: Let y denote the label for the input document $\boldsymbol{x}$. Given a neural network $f_{\boldsymbol{\eta}}$ parameterized by $\boldsymbol{\eta}$, which takes as input the latent hashing code $\boldsymbol{z}$ and predicts the document label, the supervised hashing objective to be minimized can be expressed as:

$$-\mathcal{L}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi}) + \alpha \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\big[\mathcal{L}'(y; f_{\boldsymbol{\eta}}(\boldsymbol{z}))\big], \tag{6}$$

where $\alpha > 0$ is a hyperparameter and $\mathcal{L}'(y; f_{\boldsymbol{\eta}}(\boldsymbol{z}))$ is the cross entropy loss function for label prediction.

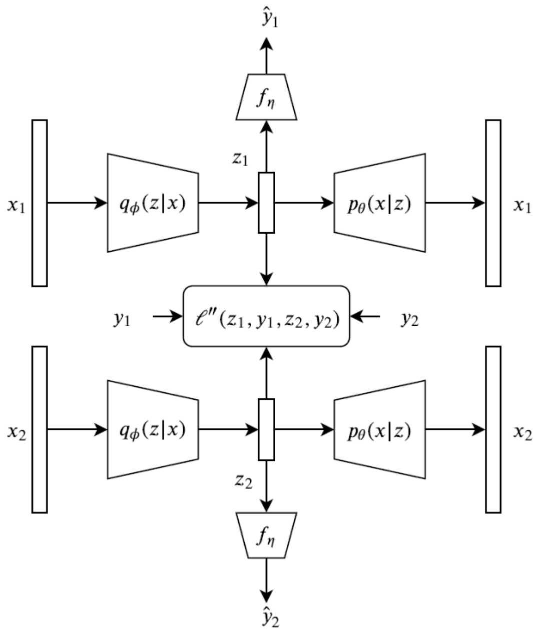

Figure 1: The graphical representation of Pairwise Supervised Hashing (PSH) model. Documents $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ each go through the encoder network ($q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$) to generate latent hashing codes $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively. Each hashing code then goes through decoder ($p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$) and classifier ($f_{\boldsymbol{\eta}}$) networks to reconstruct the input document and predict its label, respectively.

To further improve the performance of supervised hashing, we propose a pairwise supervised hashing (PSH) training framework. The core idea of PSH is to minimize the distance between latent codes of similar documents and simultaneously maximize the distance between latent codes of documents which fall into different categories. Denoting $(\boldsymbol{x}^{(1)}, y^{(1)})$ and $(\boldsymbol{x}^{(2)}, y^{(2)})$ as two randomly sampled documents with their corresponding latent codes $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$, PSH places an extra loss function as:

$$\begin{aligned}\mathcal{L}''(&\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \\ &\mathbf{1}_{y^{(1)}=y^{(2)}} d(\mathbb{E}[\boldsymbol{z}^{(1)} \mid \boldsymbol{x}^{(1)}], \mathbb{E}[\boldsymbol{z}^{(2)} \mid \boldsymbol{x}^{(2)}]) \\ &- \mathbf{1}_{y^{(1)}\neq y^{(2)}} d(\mathbb{E}[\boldsymbol{z}^{(1)} \mid \boldsymbol{x}^{(1)}], \mathbb{E}[\boldsymbol{z}^{(2)} \mid \boldsymbol{x}^{(2)}]),\end{aligned} \tag{7}$$

where $d(\cdot, \cdot)$ is a distance metric and $\mathbf{1}_S$ is the indicator function being equal to one when S is true. The final objective function for the PSH is thus

$$\begin{aligned}\mathcal{L}_{PSH}(\boldsymbol{\theta}, \boldsymbol{\phi}) = &-\big[\mathcal{L}^{(1)}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi}) + \mathcal{L}^{(2)}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi})\big] \\ &+ \beta \mathcal{L}''(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) \\ &+ \alpha \mathbb{E}_{\prod_{t=1}^{2} q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(t)} \mid \boldsymbol{x}^{(t)})} \big[\mathcal{L}'(y^{(1)}; f_{\boldsymbol{\eta}}(\boldsymbol{z}^{(1)})) \\ &+ \mathcal{L}'(y^{(2)}; f_{\boldsymbol{\eta}}(\boldsymbol{z}^{(2)}))\big],\end{aligned} \tag{8}$$

where $\mathcal{L}^{(t)}_\lambda(\boldsymbol{\theta}, \boldsymbol{\phi})$ is the ELBO for document t and $\beta > 0$ is a hyperparameter. In practice, effective hyperaparameters for PSH can be determined by cross validation. The graphical representation of PSH is shown in Figure 1.

2.3 Gradient Updates for Training

Optimizing the PSH loss function (8) is difficult, as the backpropagation algorithm cannot be applied to the discrete Bernoulli sampling layers. In this section, we first present two widely used gradient estimators for discrete latent variables. Then, we present how ARM, an unbiased gradient estimator, can be employed for backpropagation through discrete layers of our PSH framework.

2.3.1 Straight-Through Gradient Estimator

The straight-through (ST) gradient estimator (Bengio et al., 2013) simply backpropagates through a discrete sampling unit as if had been the identity function. More precisely, given the input document $\boldsymbol{x}$, first the binary latent representation is sampled as

$$\boldsymbol{z} \sim \text{Bernoulli}(\sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x}))),$$

and then the input to the decoder is calculated as

$$\boldsymbol{z}' = \text{Stop Gradient}\Big(\boldsymbol{z} - \sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x}))\Big) + \sigma(f_{\boldsymbol{\phi}}(\boldsymbol{x})),$$

where the terms inside the *Stop Gradient* operator are considered as constants in the backpropagation step (Bengio et al., 2013).

Although this is clearly a biased estimator, it is simple to implement and fast, with good performance in practice.

2.3.2 Gumbel-Softmax Gradient Estimator

The Gumbel-Softmax (GS) distribution (Jang et al., 2016; Maddison et al., 2016), a continuous distribution on the simplex, can be adopted to approximate the gradient estimates of the loss functions involving categorical samples, where parameter gradients can be computed via the reparameterization trick (Kingma and Welling, 2013). Consider an inference network architecture that for each component of latent hashing code z_k, it outputs the ratio of the probabilities of being one or zero as $\pi_k = \frac{\pi_{k1}}{\pi_{k0}}$. The binary representation of z_k can be obtained using the Gumbel-Max trick and the fact that the difference of two Gumbels is a Logistic distribution:

$$z_k = \text{unit-step}(g + \log \pi_k),$$

where g is a randomly drawn Logistic sample, which can be generated as $g = \log u - \log(1 - u)$ with $u \sim \text{Uniform}(u; 0, 1)$. In the backward pass of backpropagation, the binary random variables are replaced with continuous, differentiable variables as:

$$h_k = \frac{1}{1 + \exp(-(g + \log \pi_k)/\tau)}, \tag{9}$$

where $\tau > 0$ is the *temperature*. As the softmax temperature τ approaches zero, samples from the Gumbel-Softmax distribution become one-hot and the Gumbel-Softmax distribution becomes identical to the Bernoulli distribution.

2.3.3 Self-Control Gradient Estimator with ARM

Both ST and GS approximations lead to biased gradient estimates. To reliably derive latent codes in our PSH framework by backpropagating unbiased gradients through stochastic binary units, we employ the ARM estimator that is unbiased, exhibits low variance, and has low computational complexity (Yin and Zhou, 2019; Boluki et al., 2020; Dadaneh et al., 2020). More importantly, unlike ST and GS gradient estimators, it can be applied to non-differentiable objective functions, tailored to training discrete VAEs with the PSH loss function $\mathcal{L}_{PSH}$.

Given a vector of K binary random variables $\boldsymbol{z} = (z_1, ..., z_K)^T$, the gradient of the objective function

$$\mathcal{E}(\boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z} \sim \prod_{k=1}^{K} \text{Bernoulli}(z_k; \sigma(\psi_k))} \big[f(\boldsymbol{z}) \big]$$

with respect to $\boldsymbol{\psi} = (\psi_1, ..., \psi_K)^T$, the logits of the Bernoulli probability parameters, can be expressed as

$$\begin{aligned} \nabla_{\boldsymbol{\psi}} \mathcal{E}(\boldsymbol{\psi}) \quad &= \quad \mathbb{E}_{\boldsymbol{u} \sim \prod_{k=1}^{K} \text{Uniform}(u_k; 0,1)} \Big[\\ & \quad \Big(f(\mathbf{1}_{\boldsymbol{u} > \sigma(-\boldsymbol{\psi})}) - f(\mathbf{1}_{\boldsymbol{u} < \sigma(\boldsymbol{\psi})}) \Big) \\ \times & \quad (\boldsymbol{u} - 1/2) \Big], \end{aligned} \tag{10}$$

where $\mathbf{1}_{\boldsymbol{u} > \sigma(-\boldsymbol{\psi})} := \big(\mathbf{1}_{u_1 > \sigma(-\psi_1)}, ..., \mathbf{1}_{u > \sigma(-\psi_K)} \big)^T$, and the function $f(\cdot)$ does not need to be differentiable. Note that $\mathbf{1}_{\boldsymbol{u} > \sigma(-\boldsymbol{\psi})}$ and $\mathbf{1}_{\boldsymbol{u} < \sigma(\boldsymbol{\psi})}$ are two correlated binary vectors, which are evaluated under $f(\cdot)$ and then used to control the gradient variance. Thus we can consider ARM as a self-control gradient estimator that does not need extra baselines with learnable parameters for variance reduction.

The training steps of PSH with ARM gradient estimator are presented in Algorithm 1. It starts with sampling two mini-batches of input documents with the same size, randomly. The documents then go through the encoder network to obtain the Bernoulli logits, and the binary latent hashing codes are generated using the Bernoulli distribution. For documents in each mini-batch, the gradients of the reconstruction and KL terms with respect to the parameters of the encoder network are calculated using the ARM estimator in (10) with a single Monte Carlo sample and the closed form in (5), respectively. The gradients from the different loss terms are combined to update the parameters at each iteration.

Algorithm 1: Pairwise Supervised Hashing with ARM gradient estimator.

```
Input: Data $\{\boldsymbol{x}\}$, neural networks $f_{\boldsymbol{\phi}}$ (encoder), $f_{\boldsymbol{\theta}}$
       (decoder) and $f_{\boldsymbol{\eta}}$ (classifier), step size $\rho$.
Output: Model parameters $\boldsymbol{\phi}$, $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$.
Initialize model parameters randomly.
while not converged do
    Sample two mini-batches of data.
    for each mini-batch do
        $\boldsymbol{\psi} = f_{\boldsymbol{\phi}}(\boldsymbol{x})$
        Sample $\boldsymbol{z} \sim$ Bernoulli($\boldsymbol{\psi}$).
        Calculate $\nabla_{\boldsymbol{\phi}}$KL, the gradient of KL in (5).
        Calculate $\nabla_{\boldsymbol{\psi}} \mathcal{L}_{\lambda}^{(r)}$, the gradient of
          reconstruction term ($\mathcal{L}_{\lambda}^{(r)}$) in (4) using
          ARM (10).
        $\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\lambda}^{(r)} = \sum_k (\nabla_{\psi_k} \mathcal{L}_{\lambda}^{(r)})(\nabla_{\boldsymbol{\phi}} \psi_k)$
        $\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\lambda} = \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\lambda}^{(r)} + \lambda \nabla_{\boldsymbol{\phi}} \text{KL}$
        Calculate $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\lambda}^{(r)}$ and $\nabla_{\boldsymbol{\eta}} \mathcal{L}'$
    end
    Calculate the pairwise loss gradients $\nabla_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathcal{L}''$
    Combine the gradients to form $\nabla_{\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\eta}} \mathcal{L}_{PSH}$
    $\boldsymbol{\phi} = \boldsymbol{\phi} + \rho \nabla_{\boldsymbol{\phi}} \mathcal{L}_{PSH}$
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \rho \nabla_{\boldsymbol{\theta}} \mathcal{L}_{PSH}$
    $\boldsymbol{\eta} = \boldsymbol{\eta} + \rho \nabla_{\boldsymbol{\eta}} \mathcal{L}_{PSH}$
end
```

3 RELATED WORK

Current hashing methods can be categorized into two groups; data-dependent and data-independent. Locally sensitive hashing (LSH) (Datar et al., 2004) is a data-independent hashing method, with asymptotic theoretical properties leading to performance guarantees. LSH, however, usually requires long hashing codes to achieve satisfactory performance. To achieve more effective hashing codes, recently data-dependent machine learning methods are proposed, ranging from unsupervised and supervised to semi-supervised settings.

Unsupervised hashing methods such as Spectral Hashing (SpH) (Weiss et al., 2009), graph hashing (Liu et al., 2011), and self taught hashing (STH) (Zhang et al., 2010) attempt to extract the data properties, such as distributions and latent manifold structures to design compact codes with improved precision. Supervised hashing methods such as semantic hashing using tags and topic modeling (SHTTM) (Wang et al., 2013) and kernel-based supervised hashing (KSH) (Liu et al., 2012) attempt to leverage label/tag information for hashing function learning. A semi-supervised learning approach was also employed to design hashing functions by exploiting both labeled and unlabeled data (Wang et al., 2010).

Recently, deep learning based methods have gained attraction for the hashing problem. Variational deep semantic hashing (VDSH) (Chaidaroon and Fang, 2017) uses a VAE to learn the latent representations of documents and then uses a separate step to cast the continuous representations into binary codes. While fairly successful, this generative hashing model requires a two-stage training. Neural architecture for semantic hashing (NASH) (Shen et al., 2018) proposed to substitute the Gaussian prior in VDSH with a Bernoulli prior to tackle this problem, by using a straight-through estimator (Bengio et al., 2013) to estimate the gradient of neural network involving the binary variables.

In this work, we exploit ARM (Yin and Zhou, 2019) gradient estimator to obtain unbiased low-variance gradient updates during the training of our discrete VAE. We further propose a pairwise loss function with the discrete latent VAE to reward within-class similarity and between-class dissimilarity for supervised hashing.

4 EXPERIMENTAL RESULTS

4.1 Datasets and Baselines

We use three public benchmarks to evaluate the performance of our PSH and compare with other state-of-the-arts: *Reuters21578* and *20Newsgroups*, which are collections of news documents, as well as *TMC* from SIAM text mining competition, containing air traffic reports provided by NASA. Properties of these datasets are included in Table 1. To make a direct comparison with existing methods, we have employed the TFIDF features on these datasets.

Table 1: Properties of three datasets in the experiments.

Dataset	#documents	vocabulary size	#categories
Reuters21578	10,788	10,000	20
20Newsgroups	18,828	7,164	20
TMC	21,519	20,000	22

We evaluate the performance of our discrete latent VAEs on both unsupervised and supervised semantic hashing tasks. We consider the following unsupervised baselines for comparison: locality sensitive hashing (LSH) (Datar et al., 2004), stack restricted Boltzmann machines (S-RBM) (Salakhutdinov and Hinton, 2009), spectral hashing (SpH) (Weiss et al., 2009), self-taught hashing (STH) (Zhang et al., 2010), variational deep semantic hashing (VDSH) (Chaidaroon and Fang, 2017), and neural architecture for semantic hashing (NASH) (Shen et al., 2018).

For supervised semantic hashing, we compare the performance of PSH against a number of baselines: Supervised Hashing with Kernels (KSH) (Liu et al., 2012), Semantic Hashing using Tags and Topic Modeling (SHTTM) (Wang et al., 2013), Supervised Variational Deep Semantic Hashing (VDSH-S) (Chaidaroon and Fang, 2017), VDSH-S with document-specific latent variable (VDSH-SP) (Chaidaroon and Fang, 2017), and Supervised Neural Architecture for Semantic Hashing (NASH-DN-S) (Shen et al., 2018).

4.2 Implementation Details

For the encoder networks, we employ a fully connected neural network with two hidden layers, both with 500 units and the ReLU nonlinear activation function. We train PSH using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 5×10^{-4}. Dropout (Srivastava et al., 2014) is employed on the output of encoder networks, with the dropping rate of 0.2. To facilitate comparisons with previous methods, we set the hashing code length to 8, 16, 32, 64, or 128, respectively. For all datasets, we use a KL weight of $\lambda = 0.01$ for PSH, set the hyperparameters as $\beta = 5 \times 10^{-2}$, and start with $\alpha = 0.01$ and gradually increase its value to 0.1. The temperature of Gumbel-Softmax gradient estimator is initialized with 1, and it is gradually decreased with a decay rate of 0.96, until it reaches the minimum value of 0.1.

4.3 Evaluation Metric

To evaluate the quality of hashing codes for similarity search, we follow previous works (Shen et al., 2018; Chaidaroon and Fang, 2017) and consider each document in the test set as a query document. Specifically, the performance of different methods are measured with the *precision at 100* metric as explained in the following. In the testing phase, we first retrieve the 100 nearest documents to the query document according to the Hamming distances of their corresponding hashing codes. We then calculate the percentage of documents among the 100 retrieved ones that belong to the same label (topic) with the query document. The ratio of the number of relevant documents to the number of retrieved documents is calculated as the precision score. The precision scores are further averaged over all test (query) documents.

4.4 Results and Discussions

4.4.1 Unsupervised Hashing

To examine how our discrete latent VAE with the ARM gradient estimator affects the quality of hashing codes,

Table 2: The performances of different unsupervised hashing models on the Reuters dataset with different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
LSH	0.2802	0.3215	0.3862	0.4667	0.5194
S-RBM	0.5113	0.5740	0.6154	0.6177	0.6452
SpH	0.6080	0.6340	0.6513	0.6290	0.6045
STH	0.6616	0.7351	0.7554	0.7350	0.6986
VDSH	0.6859	0.7165	0.7753	0.7456	0.7318
NASH	**0.7113**	**0.7624**	0.7993	0.7812	0.7559
ARM-DVAE	0.6549	0.7455	**0.8086**	**0.8237**	**0.8230**

we evaluate its performance in an unsupervised scenario. More specifically, we build a binary VAE with the weighted KL regularization term on the training documents, and then use the trained encoder network to generate the binary hashing codes. To improve the performance of unsupervised hashing with VAE, we follow the procedure in Shen et al. (2018), and add a data-dependent noise to the binary hashing code before feeding it into the decoder network.

Tables 2, 3, and 4 show the performance of the proposed ARM-facilitated discrete latent VAE (hereby referred to as ARM-DVAE) and baseline models on Reuters, 20 Newsgroup and TMC datasets respectively, under the unsupervised setting, with the number of hashing bits ranging from 8 to 128. It can be observed that exploiting the unbiased and low-variance ARM gradient estimator improves the performance of unsupervised hashing in terms of the retrieval precision in the majority of cases for these datasets. In particular, for the 128-bit hashing codes, ARM-DVAE improves the performance of NASH 22% across all datasets, on average. These observations strongly support the remarkable benefit of using ARM gradient estimator to learn useful semantic hashing codes in the discrete latent VAE framework.

Table 3: The performances of different unsupervised hashing models on the 20 Newsgroup dataset with different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
LSH	0.0578	0.0597	0.0666	0.0770	0.0949
S-RBM	0.0594	0.0604	0.0533	0.0623	0.0642
SpH	0.2545	0.3200	0.3709	0.3196	0.2716
STH	0.3664	**0.5237**	**0.5860**	0.5806	0.5443
VDSH	0.3643	0.3904	0.4327	0.1731	0.0522
NASH	0.3786	0.5108	0.5671	0.5071	0.4664
ARM-DVAE	**0.3907**	0.5074	0.5787	**0.6224**	**0.6214**

Table 4: The performances of different unsupervised hashing models on TMC dataset with different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
LSH	0.4388	0.4393	0.4514	0.4553	0.4773
S-RBM	0.4846	0.5108	0.5166	0.5190	0.5137
SpH	0.5807	0.6055	0.6281	0.6143	0.5891
STH	0.3723	0.3947	0.4105	0.4181	0.4123
VDSH	0.4330	**0.6853**	0.7108	0.4410	0.5847
NASH	0.5849	0.6573	0.6921	0.6548	0.5998
ARM-DVAE	**0.6239**	0.6825	**0.7362**	**0.7541**	**0.7599**

To further examine the performance of our ARM-facilitated discrete VAE in achieving effective document hashing, we illustrate the learned latent representations of ARM-DVAE, NASH and VDSH on the 20 Newsgroup dataset in Figure 2. UMAP (McInnes et al., 2018) is used to project the 32-dimensional latent representations into a 2-dimensional space. In this figure, each data denotes a document, with each color representing one category. It can be observed that our ARM-DVAE is able to distinguish different categories of documents better than NASH with ST gradient estimator, and VDSH that binarizes normally distributed latent variables to obtain hashing codes. In particular, hashing codes from VDSH fail to form discernible clusters, confirming the advantage of using Bernoulli random variables for semantic hashing.

4.4.2 Supervised Hashing

Tables 5, 6, and 7 show the performance of the proposed and baseline models on the three datasets under the supervised setting, with the number of hashing bits ranging from 8 to 128. From these experimental results, it can be seen that for almost all datasets and hashing code lengths, the proposed PSH model outperforms all other methods in terms of retrieval precision. In particular, in 20 Newsgroup and TMC datasets, PSH with the ARM gradient estimator consistently outperforms other hashing methods by large margins. This observation signifies the role of the ARM gradient estimator to obtain effective hashing functions.

An interesting property of PSH, compared with its base discrete latent VAE models, is that it preserves the superior performance for both short and long hashing codes. For short hashing codes, this suggests the effectiveness of PSH, especially with the ARM gradient estimator, in learning useful and compact semantic latent representations of documents. For longer hashing codes, the performance of baseline methods tend to drop slightly. This phenomenon is attributed to the fact that for longer codes, the number of data points that are assigned to a certain binary code decreases exponentially. As a result, many queries may fail to return any neighbor documents (Shen et al., 2018). The results here, however, indicate that PSH does not suffer from this phenomenon, suggesting the mitigating role of the pairwise loss term.

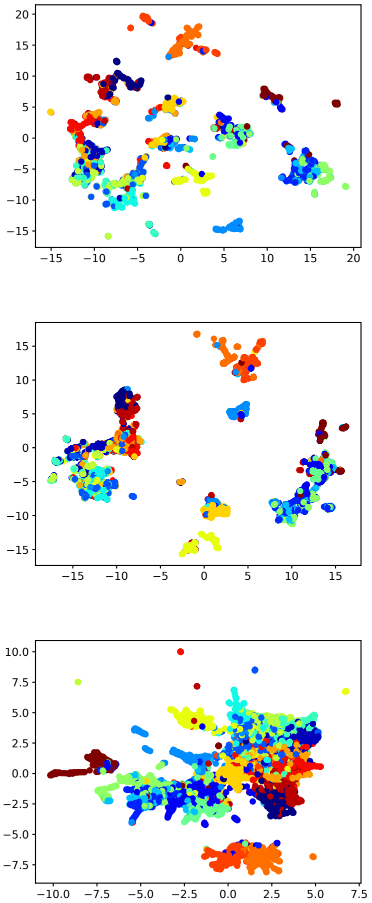

Figure 2: Visualization of the 32-dimensional latent semantic embeddings learned by ARM-DVAE (top), NASH (middle) and VDSH (bottom) for the 20 Newsgroups dataset. Each data point denotes a document, with each color representing one category. ARM-DVAE shows better separation of categories in the latent space.

4.4.3 Ablation Study

In this section, we perform ablation studies on the impacts of the pairwise loss and KL regularization terms on the performance of PSH with 32-bit hashing code. Table 8 shows the precision of PSH for document retrieval on the 20 Newsgroup dataset for various pairwise loss weight β values. We observe that discarding the pairwise loss term ($\beta = 0$) decreases the performance of the PSH in learning effective hashing codes for document retrieval. Similarly, increasing β to values higher than 0.05 degrades the performance significantly, indicating the importance of cross-validating the weight of the pairwise loss term.

Table 5: The performances of different supervised hashing models on Reuters dataset under different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
KSH	0.7840	0.8376	0.8480	0.8537	0.8620
SHTTM	0.7992	0.8520	0.8323	0.8271	0.8150
VDSH-S	0.9005	0.9121	0.9337	0.9407	0.9299
VDSH-SP	0.8890	0.9326	0.9283	0.9286	0.9395
NASH-DN-S	0.9214	0.9327	0.9380	0.9427	0.9336
PSH-GS	0.8785	**0.9604**	**0.9544**	**0.9594**	0.9528
PSH-ARM	**0.9268**	0.9458	0.9451	0.9543	**0.9569**

Table 6: The performances of different supervised hashing models on 20 Newsgroup dataset under different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
KSH	0.4257	0.5559	0.6103	0.6488	0.6638
SHTTM	0.2690	0.3235	0.2357	0.1411	0.1299
VDSH-S	0.6586	0.6791	0.7564	0.6850	0.6916
VDSH-SP	0.6609	0.6551	0.7125	0.7045	0.7117
NASH-DN-S	0.6247	0.6973	0.8069	0.8213	0.7840
PSH-GS	0.7387	0.8075	0.8274	0.8295	0.8271
PSH-ARM	**0.7507**	**0.8212**	**0.8376**	**0.8404**	**0.8432**

Table 9 illustrates the performance of PSH for document retrieval on the 20 Newsgroup dataset for various KL regularization weight λ values, indicating the sensitivity of PSH to the weight of the KL regularization term. Specifically, PSH achieves the best performance for small λ values. This observation is consistent with the literature (Zhao et al., 2017; Alemi et al., 2018), where KL weights less than one are associated with maximizing the mutual information between the observations and latent variables, hence increasing the effectiveness of hashing codes.

4.4.4 Qualitative Analysis of Semantic Information

Similar to Shen et al. (2018), we examine the nearest neighbors of some words in the word vector space learned on 20 Newsgroup dataset. We calculate the distances based on the (word embedding) matrix $E \in \mathbb{R}^{K\times|V|}$ and select top 4 of the nearest neighbors. The results for ARM-DVAE and NASH are provided in Table 10. We can see that our method places semantically-similar words closer together in the embedding space.

4.5 Computational Complexity

Our proposed framework for both supervised (PSH-ARM) and unsupervised (ARM-DVAE) semantic hashing can effectively be applied to large-scale datasets. To demonstrate this property, we apply both models on a collection of documents from the RCV1 benchmark (Lewis et al., 2004) with 100,000 training documents and 20,000 test documents. Table 11 includes the precision at 100 of PSH-ARM and ARM-DVAE on the RCV1 dataset for

Table 7: The performances of different supervised hashing models on TMC dataset under different lengths of hashing codes.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
KSH	0.6608	0.6842	0.7047	0.7175	0.7243
SHTTM	0.6299	0.6571	0.6485	0.6893	0.6474
VDSH-S	0.7387	0.7887	0.7883	0.7967	0.8018
VDSH-SP	0.7498	0.7798	0.7891	0.7888	0.7970
NASH-DN-S	0.7438	0.7946	0.7987	0.8014	0.8139
PSH-GS	0.7931	0.8189	0.8314	0.8379	0.8426
PSH-ARM	**0.8010**	**0.8329**	**0.8524**	**0.8565**	**0.8617**

Table 8: Impact of the pairwise loss term on the performance of PSH for document hashing in terms of precision.

Loss weight (β)	0	0.05	0.075	0.085	0.09
Precision	0.8280	0.8373	0.7925	0.7340	0.7154

Table 9: Impact of the KL term on the performance of PSH for document hashing in terms of precision.

KL weight (λ)	0	0.01	0.1	0.5	1	2
Precision	0.7712	0.8376	0.4954	0.4474	0.4870	0.4167

various hashing code lengths. Both methods achieve high precision values for different hashing lengths, with PSH-ARM achieving close to 0.98, indicating the effectiveness of our framework. The run-time of each epoch in the training phase for PSH-ARM and ARM-DVAE is around 0.6 and 2 minutes, respectively.

5 CONCLUSION

In this paper, we exploit Augment-REINFORCE-Merge (ARM), an unbiased, low-variance gradient estimator to build effective semantic hashing with a discrete latent VAE. Employing the ARM gradient leads to not only fast convergence, but also low negative evidence lower bounds for variational inference, thus increasing the ability to reconstruct the original word counts from the latent hashing codes. More critically, we propose PSH by adding a pairwise loss function to the base discrete VAE to reward within-class similarity and between-class dissimilarity in the supervised hashing setting. We conduct comprehensive experiments on several benchmark datasets, including the large-scale RCV1 benchmark, for both supervised and unsupervised hashing and show the superior performance of our proposed model in terms of precision for document retrieval.

Acknowledgement

The presented materials are based upon the work supported by the National Science Foundation under Grants CCF-1553281, IIS-1812641, IIS-1812699, and CCF-1934904. We also thank Texas A&M High Performance Research Computing and Texas Advanced Computing Center for providing computational resources to perform experiments in this work.

Table 10: The four nearest neighbors in the word embedding space.

Method/Word	weapons	medical	companies	book
ARM-DVAE	guns weapon violent rifles	treatment therapy medicine hospital	market company customers industry	books letters references subject
NASH	gun guns weapon armed	treatment disease drugs health	company market afford products	books english references learning

Table 11: The performances of the proposed unsupervised (ARM-DVAE) and supervised (PSH-ARM) hashing models on RCV1 dataset with different hashing code lengths.

Method	8 bits	16 bits	32 bits	64 bits	128 bits
PSH-ARM	0.9754	0.9788	0.9782	0.9737	0.9759
ARM-DVAE	0.8368	0.8899	0.8988	0.8993	0.8968

References

A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy. Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168, 2018.

A. Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

S. Boluki, R. Ardywibowo, S. Z. Dadaneh, M. Zhou, and X. Qian. Learnable Bernoulli dropout for Bayesian deep learning. *arXiv preprint arXiv:2002.05155*, 2020.

S. Chaidaroon and Y. Fang. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM, 2017.

S. Z. Dadaneh, S. Boluki, M. Zhou, and X. Qian. Arsm gradient estimator for supervised learning to rank. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3157–3161, 2020.

M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth Annual*

Symposium on Computational Geometry, pages 253–262. ACM, 2004.

A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei. Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405, 2019.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2 (1):1–19, 2006.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1–8, 2011.

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

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 441–450. ACM, 2009.

J. Ramos et al. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

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

B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, et al. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pages 285–295, 2001.

D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, R. Henao, and L. Carin. NASH: Toward end-to-end neural architecture for generative semantic hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2041–2050, 2018.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3424–3431. IEEE, 2010.

Q. Wang, D. Zhang, and L. Si. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 213–222. ACM, 2013.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.

M. Yin and M. Zhou. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *International Conference on Learning Representations*, 2019.

M. Yin, N. Ho, B. Yan, X. Qian, and M. Zhou. Probabilistic best subset selection by gradient-based optimization. *arXiv preprint arXiv:2006.06448*, 2020.

D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–25. ACM, 2010.

S. Zhao, J. Song, and S. Ermon. InfoVAE: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.