# Variational Deep Semantic Hashing for Text Documents

Suthee Chaidaroon
Department of Computer Engineering
Santa Clara University
Santa Clara, CA 95053, USA
schaidaroon@scu.edu

Yi Fang
Department of Computer Engineering
Santa Clara University
Santa Clara, CA 95053, USA
yfang@scu.edu

## ABSTRACT

As the amount of textual data has been rapidly increasing over the past decade, efficient similarity search methods have become a crucial component of large-scale information retrieval systems. A popular strategy is to represent original data samples by compact binary codes through hashing. A spectrum of machine learning methods have been utilized, but they often lack expressiveness and flexibility in modeling to learn effective representations. The recent advances of deep learning in a wide range of applications has demonstrated its capability to learn robust and powerful feature representations for complex data. Especially, deep generative models naturally combine the expressiveness of probabilistic generative models with the high capacity of deep neural networks, which is very suitable for text modeling. However, little work has leveraged the recent progress in deep learning for text hashing.

In this paper, we propose a series of novel deep document generative models for text hashing. The first proposed model is unsupervised while the second one is supervised by utilizing document labels/tags for hashing. The third model further considers document-specific factors that affect the generation of words. The probabilistic generative formulation of the proposed models provides a principled framework for model extension, uncertainty estimation, simulation, and interpretability. Based on variational inference and reparameterization, the proposed models can be interpreted as encoder-decoder deep neural networks and thus they are capable of learning complex nonlinear distributed representations of the original documents. We conduct a comprehensive set of experiments on four public testbeds. The experimental results have demonstrated the effectiveness of the proposed supervised learning models for text hashing.

## CCS CONCEPTS

**•Information systems →Information retrieval; •Computing methodologies →Neural networks; Learning latent representations;**

## KEYWORDS

Semantic hashing; Variational autoencoder; Deep learning


*SIGIR '17, Shinjuku, Tokyo, Japan*

978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080816

## 1 INTRODUCTION

The task of similarity search, also known as nearest neighbor search, proximity search, or close item search, is to find similar items given a query object [36]. It has many important information retrieval applications such as document clustering, content-based retrieval, and collaborative filtering [34]. The rapid growth of Internet has resulted in massive textual data in the recent decades. In addition to the cost of storage, searching for relevant content in gigantic databases is even more daunting. Traditional text similarity computations are conducted in the original vector space and could be prohibitive to use for large-scale corpora since these methods are involved with high cost of numerical computation in the high-dimensional spaces.

Many research efforts have been devoted to approximate similarity search that is shown to be useful for practical problems. Hashing [6, 29, 39] is an effective solution to accelerate similarity search by designing compact binary codes in a low-dimensional space so that semantically similar documents are mapped to similar codes. This approach is much more efficient in memory and computation. A binary representation of each document often only needs 4 or 8 bytes to store, and thus a large number of encoded documents can be directly loaded into the main memory. Computing similarity between two documents can be accomplished by using bitwise XOR operation which takes only one CPU instruction. A spectrum of machine learning methods have been utilized in hashing, but they often lack expressiveness and flexibility in modeling, which prevents them from learning compact and effective representations of text documents.

On the other hand, deep learning has made tremendous progress in the past decade and has demonstrated impressive successes in a variety of domains including speech recognition, computer vision, and natural language processing [19]. One of the main purposes of deep learning is to learn robust and powerful feature representations for complex data. Recently, deep generative models with variational inference [15, 28] have further boosted the expressiveness and flexibility for representation learning by integrating deep neural nets into the probabilistic generative framework. The seamless combination of generative modeling and deep learning makes them suitable for text hashing. However, to the best of our knowledge, no prior work has leveraged them for hashing tasks.

In this paper, we propose a series of novel deep document generative models for text hashing, inspired by variational autoencoder (VAE) [15, 28]. The proposed models are the marriage of deep learning and probabilistic generative models [1]. They enjoy the good properties of both learning paradigms. First, with the deep neural networks, the proposed models can learn flexible nonlinear distributed representations of the original high-dimensional documents. This allows individual codes to be fairly general and

concise but their intersection to be much more precise. For example, nonlinear distributed representations allow the topics/codes "government," "mafia," and "playboy" to combine to give very high probability to the word "Berlusconi," which is not predicted nearly as strongly by each topic/code alone.

Meanwhile, the proposed models are probabilistic generative models and thus there exists an underlying data generation process characterizing each model. The probabilistic generative formulation provides a principled framework for model extensions such as incorporating supervisory signals and adding private variables. The first proposed model is unsupervised and can be interpreted as a variant of variational autoencoder for text documents. The other two models are supervised by utilizing the document label/tag information. The prior work in the literature [37] has demonstrated that the supervisory signals are crucial to boost the performance of semantic hashing for text documents. The third model further adds a private latent variable for documents to capture the information only concerned with the documents but irrelevant to the labels, which may help remove noises from document representations. Furthermore, specific constraints can be enforced by making explicit assumptions in the models. One desirable property of hash code is to ensure the bits are uncorrelated so that the next bit cannot be predicted based on the previous bits [39]. To achieve this property, we can just assume that the latent variable has a prior distribution with independent dimensions.

In sum, the probabilistic generative formulation provides a principled framework for model extensions, interpretability, uncertainty estimation, and simulation, which are often lacking in deep learning models but useful in text hashing. The main contributions of the paper can be summarized as follow:

- We proposed a series of unsupervised and supervised deep document generative models to learn compact representations for text documents. To the best of our knowledge, this is the first work that utilizes deep generative models with variational inference for text hashing.
- The proposed models enjoy both advantages of deep learning and probabilistic generative models. They can learn complex nonlinear distributed representations of the original high-dimensional documents while providing a principled framework for probabilistic reasoning.
- We derived tractable variational lowerbounds for the proposed models and reparameterize the models so that backpropagation can be applied for efficient parameter estimation.
- We conducted a comprehensive set of experiments on four public testbeds. The experimental results demonstrate significant improvements in our supervised models over several well-known semantic hashing baselines.

## 2 RELATED WORK

### 2.1 Hashing

Due to computational and storage efficiencies of compact binary codes, hashing methods have been widely used for similarity search, which is an essential component in a variety of large-scale information retrieval systems [34, 36]. Locality-Sensitive Hashing (LSH) [2] is one of the most popular hashing methods with interesting asymptotic theoretical properties leading to performance guarantees. While LSH is a data-independent hashing method, many hashing methods have been recently proposed to leverage machine learning techniques with the goal of learning data-dependent hash functions, ranging from unsupervised and supervised to semi-supervised settings. Unsupervised hashing methods attempt to integrate the data properties, such as distributions and manifold structures to design compact hash codes with improved accuracy. For instance, Spectral Hashing (SpH) [39] explores the data distribution by preserving the similarity between documents by forcing the balanced and uncorrelated constraints into the learned codes, which can be viewed as an extension of spectral clustering [26]. Graph hashing [22] utilizes the underlying manifold structure of data captured by a graph representation. Self Taught Hashing (STH) [42] is the state-of-the-art hashing method by decomposing the learning procedure into two steps: generating binary code and learning hash function.

Supervised hashing methods attempt to leverage label/tag information for hash function learning. It has attracted more and more attention in recent years. For example, Wang et al. [37] propose Semantic Hashing using Tags and Topic Modeling (SHTTM) to incorporate tags to obtain more effective hashing codes via a matrix factorization formulation. To utilize the pairwise supervision information in the hash function learning, Kernel-Based Supervised Hashing (KSH) proposed in [21] used a pairwise relationship between samples to achieve high-quality hashing. Binary Reconstructive Embedding (BRE) [16] was proposed to learn hash functions by minimizing the reconstructed error between the metric space and Hamming space. Moreover, there are also several works using the ranking order information to design hash functions. Ranking-based Supervised Hashing (RSH) [35] was proposed to leverage listwise supervision into the hash function learning framework. Semi-supervised learning paradigm was also employed to design hash functions by using both labeled and unlabeled data [33]. The hashing-code learning problem is essentially a discrete optimization problem which is difficult to solve. Most existing supervised hashing methods try to solve a relaxed continuous optimization problem and then threshold the continuous representation to obtain a binary code. Abundant related work, especially on image hashing, exists in the literature. Two recent surveys [34, 36] provide a comprehensive literature review.

### 2.2 Deep Learning

Deep learning has drawn increasing attention and research efforts in a variety of artificial intelligence areas including speech recognition, computer vision, and natural language processing. Since one main purpose of deep learning is to learn robust and powerful feature representations for complex data, it is very natural to leverage deep learning for exploring compact hash codes which can be regarded as binary representations of data. Most of the related work has focused on image data [4, 17, 20, 40] rather than text documents probably due to the effectiveness of the convolution neural networks (CNNs) to learn good low-dimensional representations of images. The typical deep learning architectures for hash function learning consist of CNNs layers for representation learning and hash function layers which then transform the representation to

supervisory signals. The loss functions could be pointwise [20], pairwise [4], or listwise [17].

Some recent works have applied deep learning for several IR tasks such as ad-hoc retrieval [10], web search [12], and ranking pairs of short texts [30]. However, very few has investigated deep learning for text hashing. The representative work is semantic hashing [29]. It builds a stack of restricted Boltzmann machines (RBMs) [11] to discover hidden binary units which can model input text data (i.e., word-count vectors). After learning a multilayer RBM through pretraining and fine tuning on a collection of documents, the hash code of any document is acquired by simply thresholding the output of the deepest layer. A recent work [41] exploited convolutional neural network for text hashing, which relies on external features such as the GloVe word embeddings to construct text representations.

Recently, deep generative models have made impressive progress with the introduction of the variational autoencoders (VAEs) [15, 28] and Generative Adversarial Networks (GANs) [9]. VAEs are especially an appealing framework for generative modeling by coupling the approach of variational inference [32] with deep learning. As a result, they enjoy the advantages of both deep learning and probabilistic graphical models. Deep generative models parameterized by neural networks have achieved state-of-the-art performance in unsupervised and supervised learning [14, 15, 25]. To the best of our knowledge, our proposed models are the first work that utilizes variational inference with deep learning for text hashing. It is worth pointing out that both semantic hashing with stacked RBMs [29] and our models are deep generative models, but the former is undirected graphical models, and the latter is directed models. The underlying generative process of directed probabilistic models makes them easy to interpret and extend. The proposed models are very scalable since they are trained as deep neural networks by efficient backpropagation, while the stacked RBMs are often much harder to train [11].

## 3 VARIATIONAL DEEP SEMANTIC HASHING

This section presents three novel deep document generative models to learn low-dimensional semantic representations of documents for text hashing. In Section 3.1, we introduce the basic model which is essentially a variational autoencoder for text modeling. Section 3.2 extends the model to utilize label information to learn a more sensible representation. Section 3.3 further incorporates document private variables to model document-specific information. Based on the variational inference, all the three models can be viewed as having an encoder-decoder neural network architecture where the encoder compresses a high-dimensional document to a compact latent semantic vector and the decoder reconstructs the document (or the labels). Section 3.4 discusses two thresholding methods to convert the continues latent vector to a binary code for text hashing.

### 3.1 Unsupervised Learning (VDSH)

In this section, we present the basic variational deep semantic hashing (VDSH) model for the unsupervised learning setting. VDSH is a probabilistic generative model of text which aims to extract a continuous low-dimensional semantic representation $s \in \mathrm{R}^K$ for each document. Let $d \in \mathrm{R}^V$ be the bag-of-words representation of a document and $w_i \in \{0,1\}^V$ be the one-hot vector representation of the $i^{th}$ word of the document where $V$ is the vocabulary size. $d$ could be represented by different term weighting schemes such as binary, TF, and TFIDF [24]. The document generative process can be described as follows:

For each document $d$,
- Draw a latent semantic vector $s \sim P(s)$ where $P(s) = \mathrm{N}(0, I)$ is the standard Gaussian distribution.
- For the $i^{th}$ word in the document,
  Draw $w_i \sim P(w_i \mid f(s;\theta))$.

The conditional probability over words $w_i$ is modelled by multinomial logistic regression and shared across documents as below:

$$P(w_i \mid f(s;\theta)) = \frac{\exp(w_i^T f(s;\theta))}{\sum_{j=1}^{V} \exp(w_j^T f(s;\theta))} \quad (1)$$

While $P(s)$ is a simple Gaussian distribution, any distribution can be generated by mapping the simple Gaussian through a sufficiently complicated function [3]. Thus, $f(s;\theta))$ is such a highly flexible function approximator usually a neural network. In other words, we can learn a function which maps our independent, normally-distributed $s$ values to whatever latent semantic variables might be needed for the model, and then generate the word $w_i$. However, introducing a highly nonlinear mapping from $s$ to $w_i$ results in intractable data likelihood $\int_s P(d \mid s)P(s)ds$ and thus intractable posterior distribution $P(s \mid d)$ [15]. Similar to VAE, we use an approximation $Q(s \mid d;\phi)$ for the true posterior distribution. By applying the variational inference principle [32], we can obtain the following tractable lowerbound of the document log likelihood (see [15] and Appendix):

$$L_1 = E_Q\left\{\sum_{i=1}^{N} \log P(w_i \mid f(s;\theta))\right\} - D_{KL}(Q(s \mid d;\phi) \,\|\, P(s)) \quad (2)$$

where $N$ is the number of words in the document and $D_{KL}(\,\|\,)$ is the Kullback-Leibler (KL) divergence between the approximate posterior distribution $Q(s \mid d;\phi)$ and the prior $P(s)$. The variational distribution $Q(s \mid d;\phi)$ acts as a proxy to the true posterior $P(s \mid d)$. To enable a high capacity, it is assumed to be a Gaussian $\mathrm{N}(\mu, \mathrm{diag}(\sigma^2))$ whose mean $\mu$ and variance $\sigma^2$ are the output of a highly nonlinear function of $d$ denoted as $\partial(d;\phi)$ parameterized by $\phi$, once again typically a neural network.

In training, the variational lowerbound in Eqn.(2) is maximized with respect to the model parameters. Since $P(s)$ is a standard Gaussian prior, the KL Divergence $D_{KL}(Q(s \mid d;\phi) \,\|\, P(s))$ in Eqn.(2) can be computed analytically. The first term $E_Q$ can be viewed as an expected negative reconstruction error of the words in the document and it can be computed based on the Monte Carlo estimation [8].

Based on Eqn.(2), we can interpret VDSH as a variational autoencoder with discrete output: a feedforward neural network encoder $Q(s \mid d;\phi)$ compresses document representations into continuous hidden vectors, i.e., $d \to s$; a softmax decoder $\prod_{i=1}^{N} P(w_i \mid f(s;\theta))$ reconstructs the documents by independently generating the words $s \to \{w_i\}_{i=1}^{N}$. Figure 1(a) illustrates the architecture of VDSH. In

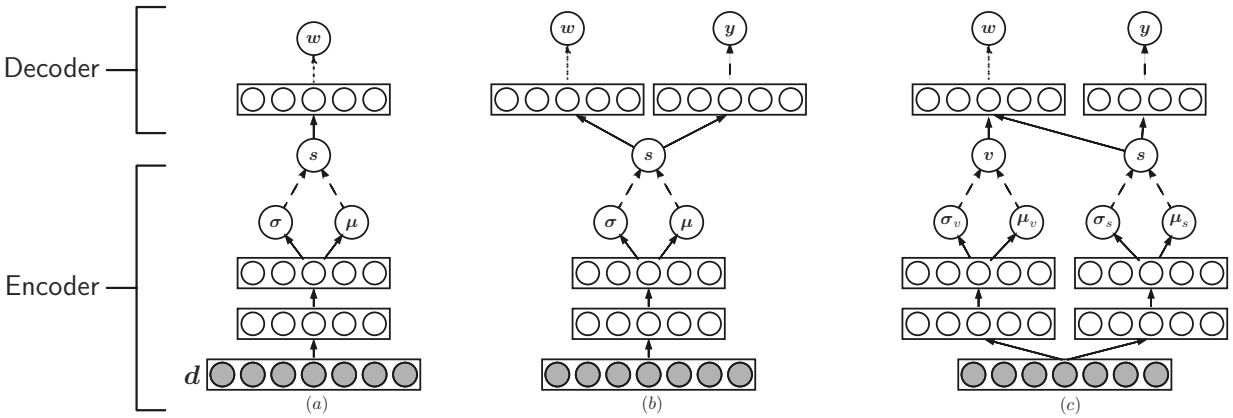



**Figure 1: Architectures of (a) VDSH, (b) VDSH-S, and (c) VDSH-SP. The dashed line represents a stochastic layer.**

the experiments, we use the following specific architecture for the encoder and decoder.

$$\text{Encoder} \quad Q(s|d;\phi)): \qquad \text{Decoder} \quad P(w_i|f(s;\theta)):$$

$$t_1 = \text{ReLU}(W_1 d + b_1) \qquad c_i = \exp(s^T G w_i + b_{w_i})$$

$$t_2 = \text{ReLU}(W_2 t_1 + b_2) \qquad P(w_i|s) = \frac{c_i}{\sum_{k=1}^{V} c_k}$$

$$\mu = W_3 t_2 + b_3 \qquad P(d|s) = \prod_{i=1}^{N} P(w_i|s)$$

$$\log\sigma = W_4 t_2 + b_4$$

$$s \quad \mathcal{N}(\mu(d), \text{diag}(\sigma^2(d)))$$

This architecture is similar to the one presented in VAE [28] except that VDSH has the softmax layer to model discrete words while VAE is proposed to model images as continuous output. Here, the encoder has two Rectified Linear Unit (ReLU) [8] layers. ReLU generally does not face gradient vanishing problem as with other activation functions. Also, it has been shown that deep neural networks can be trained efficiently using ReLU even without pre-training [8].

In this architecture, there is a stochastic layer which is to sample $s$ from a Gaussian distribution $\mathcal{N}(\mu(d), \text{diag}(\sigma^2(d)))$, as represented by the dashed lines in the middle of the networks in Figure 1. Backpropagation cannot handle stochastic layer within the network. In practice, we can leverage the "location-scale" property of Gaussian distribution, and use the reparameterization trick [15] to turn the stochastic layer of $s$ to be deterministic. As a result, the encoder $Q(s|d;\phi)$ and decoder $P(w_i|f(s;\theta))$ form an end-to-end neural network and are then trained jointly by maximizing the variational lowerbound in Eqn.(2) with respect to their parameters by the standard backpropagation algorithm [8].

## 3.2 Supervised Learning (VDSH-S)

In many real-world applications, documents are often associated with labels or tags which may provide useful guidance in learning effective hashing codes. Document content similarity in the original bag-of-word space may not fully reflect the semantic relationship between documents. For example, two documents in the same category may have low document content similarity due to the vocabulary gap, while their semantic similarity could be high. In this section, we extend VDSH to the supervised setting with the new model denoted as VDSH-S. The probabilistic generative process of a document with labels is as follows:

For each document $d$,

- Draw a latent semantic vector $s \sim P(s)$ where $P(s) = \mathcal{N}(0; I)$ is the standard Gaussian distribution.
- For the $i^{th}$ word in the document, Draw $w_i \sim P(w_i|f(s;\theta))$.
- For the $j^{th}$ label in the label set, Draw $y_j \sim P(y|f(s;\tau))$.

where $y_j \in \{0,1\}^L$ is the one-hot representation of the label $j$ in the label set and $L$ is the total number of possible labels (the size of the label set). Let us use $Y \in \{0,1\}^L$ represent the bag-of-labels of the document (i.e., if the document has label $j$, the $j^{th}$ dimension of $Y$ is 1; otherwise, it is 0). VDSH-S assumes that both words and labels are generated based on the same latent semantic vecotor.

We assume a general multi-label classification setting where each document could have multiple labels/tags. $P(y_j|f(s;\tau))$ can be modeled by the logistic function as follows:

$$P(y_j|f(s;\tau)) = \frac{1}{1+\exp(-y_j^T f(s;\tau))} \tag{3}$$

Similar to VDSH, $f(s;\tau)$ is parameterized by a neural network with the parameter $\tau$ so that we can learn an effective mapping from the latent semantic vector to the labels. The lowerbound of the data log likelihood can be similarly derived and shown as follows:

$$\mathcal{L}_2 = \mathbb{E}_Q\Big[\sum_{i=1}^{N} \log P(w_i|f(s;\theta)) + \sum_{j=1}^{K} \log P(y_j|f(s;\tau))\Big] - D_{KL}(Q(s|d;Y;\phi) \,\|\, P(s)) \tag{4}$$

Compared to Eqn.(2) in VDSH, this lowerbound has an extra term, $\mathbb{E}_Q\big[\sum_{j=1}^{L} \log P(y_j|f(s;\tau))\big]$, which can be computed in a similar way with $\mathbb{E}_Q\big[\sum_{i=1}^{N} \log P(w_i|f(s;\theta))\big]$ in Eqn.(2), by using the Monte Carlo estimation. In addition, we can drop the dependence on variable $Y$ in the variational distribution $Q(s|d;Y;\phi)$ since we may not have the label information available for new documents.

The architecture of the VDSH-S model is shown in Figure 1(b). It consists of a feedforward neural network encoder of a document $d \to s$ and a decoder of the words and labels of the document $s \to \{w_i\}_{i=1}^{N}; \{y_j\}_{j=1}^{L}$. It is worth pointing out that the labels still affect the learning of latent semantic vector by their presence in the decoder despite their absence in the encoder. By using the reparameterization trick, the model becomes a deterministic deep neural network and the lowerbound in Eqn.(4) can be maximized by backpropagation (see Appendix).

### 3.3 Document-specific Modeling (VDSH-SP)

VDSH-S assumes both document and labels are generated by the same latent semantic vector $s$. In some cases, this assumption may be restrictive. For example, the original document may contain information that is irrelevant to the labels. It could be difficult to find a common representation for both documents and labels. This observation motivates us to introduce a document private variable $v$, which is not shared by the labels $Y$. The generative process is described as follows:

For each document $d$,
- Draw a latent semantic vector $s \quad P(s)$ where $P(s) = \mathcal{N}(0; I)$ is the standard Gaussian distribution.
- Draw a latent private vector $v \quad P(v)$ where $P(v) = \mathcal{N}(0; I)$ is the standard Gaussian distribution.
- For the $i^{th}$ word in the document, Draw $w_i \quad P(w_i | f(s + v; \theta))$.
- For the $j^{th}$ label in the label set, Draw $y_j \quad P(y_j | f(s; \tau))$.

As we can see, $s$ models the shared information between document and labels while $v$ only contains the document-specific information. We can view adding private variables as removing the noise from the original content that is irrelevant to the labels. With the added private variable, we denote this model as VDSH-SP. The tractable variational lowerbound of data likelihood can be derived as follows:

$$L_3 = E_Q \Big\{ \sum_{i=1}^{N} \log P(w_i | f(s + v; \theta)) + \sum_{j=1}^{K} \log P(y_j | f(s; \tau)) \Big\} - D_{KL}(Q(s|d; \phi) \,\|\, P(s)) - D_{KL}(Q(v|d; \phi) \,\|\, P(v)) \quad (5)$$

Similar to the other two models, VDSH-SP can be viewed as a deep neural network by applying variational inference and reparametrization. The architecture is shown in Figure 1(c). The Appendix contains the detailed derivations of the model.

### 3.4 Binary Hash Code

Once a VDSH model has been trained, we can generate a compact continuous representation for any new document $d_{new}$ by the encoder function $\mu_{new} = \partial(d_{new}; \phi)$, which is the mean of the distribution $Q(s|d; \phi)$. The binary hashing code can then be obtained by thresholding $\mu_{new}$. The most common method of thresholding for binary code is to take the median value of the latent semantic vector $\mu$ in the training data [37]. The rationale is based on the maximum entropy principle for efficiency which yields balanced partitioning of the whole dataset [39]. Thus, we set the threshold for binarizing the $p^{th}$ bit to be the median of the $p^{th}$ dimension of $s$ in the training data. If the $p^{th}$ bit of document latent semantic vector $\mu_{new}$ is larger than the median, the $p^{th}$ binary code is set to 1; otherwise, it is set to -1. Another popular thresholding method is to use the Sign function on $\mu_{new}$, i.e., if the $p^{th}$ bit of $\mu_{new}$ is nonnegative, the corresponding code is 1; otherwise, it is -1. Since the prior distribution of the latent semantic vector is zero mean, the Sign function is also a reasonable choice. We use the median thresholding as the default method in our experiments, while also investigate the Sign function in Section 5.3.

### 3.5 Discussions

The computational complexity of VDSH for a training document is $O(BD^2 + DSV)$. Here, $O(BK^2)$ is the cost of the encoder, where $B$ is the number of the layers in the encoder network and $D$ is the average dimension of these layers. $O(DNV)$ is the cost of the decoder, where $S$ is the average length of the documents and $V$ is the vocabulary size. The computational complexity of VDSH-S and VDSH-SP is $O(BD^2 + DS(V + L))$ where $L$ is the size of the label set. The computational cost of the proposed models is at the same level as the deterministic autoencoder. Model learning could be quite efficient since the computations of all the models can be parallelized in GPUs, and only one sample is required during the training process.

The proposed deep generative model has a few desirable properties for text hashing. First of all, it has the capacity of deep neural networks to learn sophisticated semantic representations for text documents. Moreover, being generative models brings huge advantages over other deep learning models such as Convolutional Neural Network (CNN) because the underlying document generative process makes the model assumptions explicit. For example, as shown in [39], it is desirable to have independent feature dimensions in hash codes. To achieve this, our models just need to assume the latent semantic vector is drawn from a prior distribution with independent dimensions (e.g., standard Gaussian). The probabilistic approach also provides a principled framework for model extensions as evident in VDSH-S and VDSH-SP. Furthermore, instead of learning a particular latent semantic vector, our models learn probability distributions of the semantic vector. This can be viewed as finding a region instead of a fixed point in the latent space for document representation, which leads to more robust models. Compared with other deep generative models such as stacked RBMs and GANs, our models are computationally tractable and stable and can be estimated by the efficient backpropagation algorithm.

## 4 EXPERIMENTAL SETUP

### 4.1 Data Collections

We use the following four public document collections for evaluation. 1) *Reuters Corpus Volume I (RCV1).* It is a large collection of manually labeled 800,000 newswire stories provided by Reuters. There are totally 103 classes. We use the full-topics version available at the LIBSVM website[1]. 2) *Reuters21578*[2]. A widely used text corpus for text classification. This collection has 10,788 documents with 90 categories and 7,164 unique words. 3) *20Newsgroups*[3].

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/multilabel.html
[2] http://www.nltk.org/book/ch02.html
[3] http://ana.cachopo.org/datasets-for-single-label-text-categorization

| | *RCV1* | | | | | *Reuters* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Methods | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
| LSH [2] | 0.4180 | 0.4352 | 0.4716 | 0.5214 | 0.5877 | 0.2802 | 0.3215 | 0.3862 | 0.4667 | 0.5194 |
| SpH [39] | 0.5093 | 0.7121 | 0.7475 | 0.7559 | 0.7423 | 0.6080 | 0.6340 | 0.6513 | 0.6290 | 0.6045 |
| STHs [42] | 0.3975 | 0.4898 | 0.5592 | 0.5945 | 0.5946 | 0.6616 | 0.7351 | 0.7554 | 0.7350 | 0.6986 |
| Stacked RBMs [29] | 0.5106 | 0.5743 | 0.6130 | 0.6463 | 0.6531 | 0.5113 | 0.5740 | 0.6154 | 0.6177 | 0.6452 |
| KSH [21] | 0.9126 | 0.9146 | 0.9221 | 0.9333 | 0.9350 | 0.7840 | 0.8376 | 0.8480 | 0.8537 | 0.8620 |
| SHTTM [37] | 0.8820 | 0.9038 | 0.9258 | 0.9459 | 0.9447 | 0.7992 | 0.8520 | 0.8323 | 0.8271 | 0.8150 |
| **VDSH** | 0.7976 | 0.7944 | 0.8481 | 0.8951 | 0.8444 | 0.6859 | 0.7165 | 0.7753 | 0.7456 | 0.7318 |
| **VDSH-S** | 0.9652$\gamma$ | 0.9749$\gamma$ | **0.9801**$\gamma$ | 0.9804$\gamma$ | **0.9800**$\gamma$ | **0.9005**$\gamma$ | 0.9121$\gamma$ | **0.9337**$\gamma$ | **0.9407**$\gamma$ | 0.9299$\gamma$ |
| **VDSH-SP** | **0.9666**$\gamma$ | **0.9757**$\gamma$ | 0.9788$\gamma$ | **0.9805**$\gamma$ | 0.9794$\gamma$ | 0.8890$\gamma$ | **0.9326**$\gamma$ | 0.9283$\gamma$ | 0.9286$\gamma$ | **0.9395**$\gamma$ |
| | *20Newsgroups* | | | | | *TMC* | | | | |
| Methods | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
| LSH [2] | 0.0578 | 0.0597 | 0.0666 | 0.0770 | 0.0949 | 0.4388 | 0.4393 | 0.4514 | 0.4553 | 0.4773 |
| SpH [39] | 0.2545 | 0.3200 | 0.3709 | 0.3196 | 0.2716 | 0.5807 | 0.6055 | 0.6281 | 0.6143 | 0.5891 |
| STH [42] | 0.3664 | 0.5237 | 0.5860 | 0.5806 | 0.5443 | 0.3723 | 0.3947 | 0.4105 | 0.4181 | 0.4123 |
| Stacked RBMs [29] | 0.0594 | 0.0604 | 0.0533 | 0.0623 | 0.0642 | 0.4846 | 0.5108 | 0.5166 | 0.5190 | 0.5137 |
| KSH [21] | 0.4257 | 0.5559 | 0.6103 | 0.6488 | 0.6638 | 0.6608 | 0.6842 | 0.7047 | 0.7175 | 0.7243 |
| SHTTM [37] | 0.2690 | 0.3235 | 0.2357 | 0.1411 | 0.1299 | 0.6299 | 0.6571 | 0.6485 | 0.6893 | 0.6474 |
| **VDSH** | 0.3643 | 0.3904 | 0.4327 | 0.1731 | 0.0522 | 0.4330 | 0.6853 | 0.7108 | 0.4410 | 0.5847 |
| **VDSH-S** | 0.6586$\gamma$ | **0.6791**$\gamma$ | **0.7564**$\gamma$ | 0.6850$\gamma$ | 0.6916$\gamma$ | 0.7387$\gamma$ | **0.7887**$\gamma$ | 0.7883$\gamma$ | **0.7967**$\gamma$ | **0.8018**$\gamma$ |
| **VDSH-SP** | **0.6609**$\gamma$ | 0.6551$\gamma$ | 0.7125$\gamma$ | **0.7045**$\gamma$ | **0.7117**$\gamma$ | **0.7498**$\gamma$ | 0.7798$\gamma$ | **0.7891**$\gamma$ | 0.7888$\gamma$ | 0.7970$\gamma$ |

**Table 1: Precision of the top 100 retrieved documents on four datasets with different numbers of hashing bits. The bold font denotes the best result at that number of bits. $\gamma$ denotes the improvement over the best result of the baselines is statistically significant based on the paired t-test (p-value $<$ 0.01).**

This dataset is a collection of 18,828 newsgroup posts, partitioned (nearly) evenly across 20 different newsgroups/categories. It has become a popular dataset for experiments in text applications of machine learning techniques. 4) *TMC*[4]. This dataset contains the air traffic reports provided by NASA and was used as part of the SIAM text mining competition. It has 22 labels, 21,519 training documents, 3,498 test documents, and 3,498 documents for the validation set. All the datasets are multi-label except *20Newsgroups*.

Each dataset was split into three subsets with roughly 80% for training, 10% for validation, and 10% for test. The training data is used to learn the mapping from document to hash code. Each document in the test set is used to retrieve similar documents based on the mapping, and the results are evaluated. The validation set is used to choose the hyperparameters. We removed the stopwords using SMART's list of 571 stopwords[5]. No stemming was performed. We use TFIDF [24] as the default term weighting scheme for the raw document representation (i.e., $\boldsymbol{d}$). We experiment with other term weighting schemes in Section 5.4.

## 4.2 Baselines and Settings

We compare the proposed models with the following six competitive baselines which have been extensively used for text hashing in the prior work [37]: Locality Sensitive Hashing (LSH)[6] [2], Spectral Hashing (SpH)[7] [39], Self-taught Hashing (STH)[8] [42], Stacked Restricted Boltzmann Machines (Stacked RBMs) [29], Supervised Hashing with Kernels (KSH) [21], and Semantic Hashing using Tags and Topic Modeling (SHTTM) [37]. We used the validation dataset to choose the hyperparameters for the baselines.

For our proposed models, we adopt the method in [7] for weight initialization. The Adam optimizer [13] with the step size 0.001 is used due to its fast convergence. Following the practice in [38], we use the dropout technique [31] with the keep probability of 0.8 in training to alleviate overfitting. The number of hidden nodes of the models is 1,500 for RCV1 and 1,000 for the other three smaller datasets. All the experiments were conducted on a server with 2 Intel E5-2630 CPUs and 4 GeForce GTX TITAN X GPUs. The proposed deep models were implemented on the Tensorflow[9] platform. For the VDSH model on the Reuters21578, 20Newsgroups, and TMC datasets, each epoch takes about 60 seconds, and each run takes 30 epochs to converge. For RCV1, it takes about 3,600 seconds per epoch and needs fewer epochs (about 15) to get satisfactory performance. Since RCV1 is much larger than the other three datasets, this shows that the proposed models are quite scalable. VDSH-S and VDSH-SP take slightly more time to train than VDSH does (about 40 minutes each on Reuters21578, 20Newsgroups, and TMC, and 20 hours on RCV1).

## 4.3 Evaluation Metrics

To evaluate the effectiveness of hashing code in similarity search, each document in the test set is used as a query document to search for similar documents based on the Hamming distance (i.e., number

[4]https://catalog.data.gov/dataset/siam-2007-text-mining-competition-dataset
[5]http://www.lextek.com/manuals/onix/stopwords2.html
[6]http://pixelogik.github.io/NearPy/
[7]http://www.cs.huji.ac.il/ yweiss/SpectralHashing/
[8]http://www.dcs.bbk.ac.uk/ dell/publications/dellzhang_sigir2010/sth_v1.zip
[9]https://www.tensorflow.org/

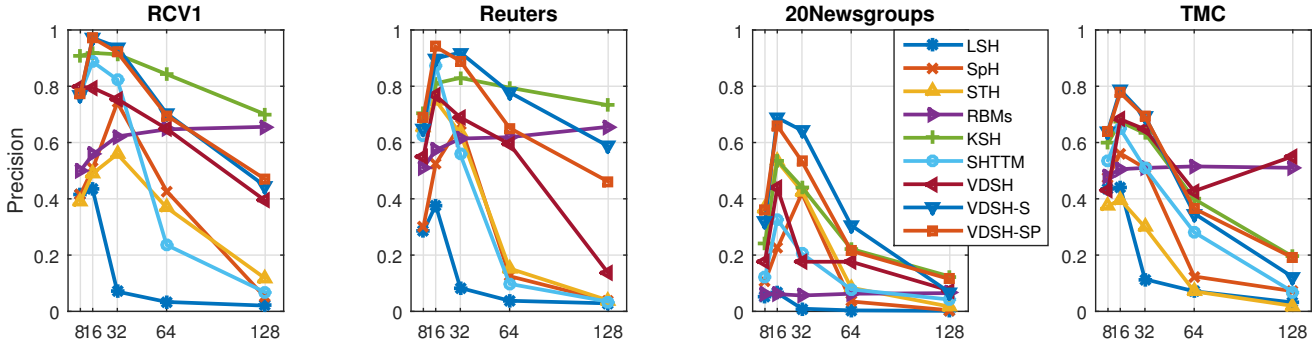



**Figure 2: The Precision within the Hamming distance of 2 on four datasets with different hashing bits.**

of different bits) between their hashing codes. Following the prior work in text hashing [37], the performance is measured by Precision, as the ratio of the number of retrieved relevant documents to the number of all retrieved documents. The results are averaged over all the test documents.

There exist various ways to determine whether a retrieved document is relevant to the given query document. In SpH [39], the $K$ closest documents in the original feature space are considered as the relevant documents. This metric is not desirable since the similarity in the original feature space may not well reflect the document semantic similarity. Also, it is hard to determine a suitable $K$ for the cutoff threshold. Instead, we adopt the methodology used in KSH [33], SHTTM [37] and other prior work [33], that is a retrieved document that shares any common test label with the query document is regarded as a relevant document.

## 5 EXPERIMENTAL RESULTS

### 5.1 Baseline Comparison

Table 1 shows the results of different methods over various numbers of bits on the four testbeds. We have several observations from the results. First of all, the best results at different bits are all achieved by VDSH-S or VDSH-SP. They consistently yield better results than all the baselines across all the different numbers of bits. All the improvements over the baselines are statistically significant based on the paired t-test (p-value $<$ 0:01). VDSH-S and VDSH-SP produce comparable results between them. Adding private variables does not always help because it increases the model flexibility which may lead to overfitting to the training data. This probably explains why VDSH-SP generally yield better performance when the number of bits is 8 which corresponds to a simpler model.

Secondly, the supervised hashing techniques (i.e., VDSH-S, VDSH-SP, KSH) outperform the unsupervised methods on the four datasets across all the different bits. These results demonstrate the importance of utilizing supervisory signals for text hashing. However, the unsupervised model, STHs, outperforms SHTTM on the original 20 categories Newsgroups. One possible explanation is that SHTTM depends on LDA to learn an initial representation. But many categories in Newsgroup are correlated, LDA could assign similar topics to documents from related categories (i.e. Christian, Religion). Hence SHTTM may not effectively distinguish two related categories. Evidently, SHTTM and KSH deliver comparable results except on the 20Newsgroups testbed. It is worth noting that there exist substantial gaps between the supervised and unsupervised proposed models (VDSH-S and VDSH-SP vs VDSH) across all the datasets and configurations. The label information seems remarkably useful for guiding the deep generative models to learn effective representations. This is probably due to the high capacity of the neural network component which can learn subtle patterns from supervisory signals when available.

Thirdly, the performance does not always improve when the number of bits increases. This pattern seems quite consistent across all the compared methods and it is likely the result of model overfitting, which suggests that using a long hash code is not always helpful especially when training data is limited. Last but not least, the testbeds may affect the model performance. All the best results are obtained on the RCV1 dataset whose size is much larger than the other testbeds. These results illustrate the importance of using a large amount of data to train text hashing models.

It is worth noting that some of the baseline results are different from what were reported in the prior work. This is due to the data preprocessing. For example, [37] combined some categories in 20Newsgroup to form 6 broader categories in their experiments while we use all the original 20 categories for evaluation. [42] focused on single-label documents by discarding the documents appearing in more than one category while we use all the documents in the corpus.

### 5.2 Retrieval with Fixed Hamming Distance

In practice, IR systems may retrieve similar documents in a large corpus within a fixed Hamming distance radius to the query document. In this section, we evaluate the precision for the Hamming radius of 2. Figure 2 shows the results on four datasets with different numbers of hashing bits. We can see that the overall best performance among all nine hashing methods on each dataset is achieved by either VDSH-S or VDSH-SP at the 16-bit. In general, the precision of most of the methods decreases when the number of hashing bits increases from 32 to 128. This may be due to the fact that when using longer hashing bits, the Hamming space becomes increasingly sparse and very few documents fall within the Hamming distance of 2, resulting in more queries with precision 0. Similar behavior is also observed in the prior work such as KSH [21] and SHTTM [37]. A notable exception is Stacked RBMs whose

| | RCV1 | | Reuters | |
|---|---|---|---|---|
| | Median | Sign | Median | Sign |
| VDSH | 0.8481 | 0.8514 | 0.7753 | 0.7851 |
| VDSH-S | 0.9801 | 0.9804 | 0.9337 | 0.9284 |
| VDSH-SP | 0.9788 | 0.9794 | 0.9283 | 0.9346 |
| | 20Newsgroups | | TMC | |
| | Median | Sign | Median | Sign |
| VDSH | 0.4354 | 0.4267 | 0.7108 | 0.7162 |
| VDSH-S | 0.7564 | 0.7563 | 0.7883 | 0.7879 |
| VDSH-SP | 0.6913 | 0.6574 | 0.7891 | 0.7761 |

**Table 2: Precision@100 of using different thresholding functions (Median vs Sign) for the proposed models on four testbeds with the 32-bit hash code**

performance is quite stable across different numbers of bits while lags behind the best performers.

## 5.3 Effect of Thresholding

Thresholding is an important step in hashing to transform a continuous document representation to a binary code. We investigate two popular thresholding functions: Median and Sign, which are introduced in Section 3.4. Table 2 contains the precision results of the proposed models with the 32-bit hash code on the four datasets. As we can see, the two thresholding functions generate quite similar results and their differences are not statistically significant, which indicates all the proposed models, whether being unsupervised or supervised, are not sensitive to the thresholding methods.

## 5.4 Effect of Term Weighting Schemes

In this section, we investigate the effect of term weighting schemes on the performance of the proposed models. Different term weights result in different bag-of-word representations of $\boldsymbol{d}$ as the input to the neural network. Specifically, we experiment with three term weighting representations for documents: Binary, Term Frequency (TF), Term Frequency and Inverse Document Frequency (TFIDF) [24]. Figure 3 illustrates the results of the proposed models with the 32-bit hash code on the four datasets. As we can see, the proposed models generally are not very sensitive to the underlying term weighting schemes. The TFIDF weighting always gives the best performance on all the four datasets. The improvement is more noticeable with VDSH-S and VDSH-SP on 20Newsgroups. The results indicate more sophisticated weighting schemes may capture more information about the original documents and thus lead to better hashing results. One the other hand, all the three models yield quite stable results on RCV1, which suggests that a large-scale dataset may help alleviate the shortcomings of the basic term weighting schemes.

## 5.5 Qualitative Analysis

In this section, we visualize the low-dimensional representations of the documents and see whether they can preserve the semantics of the original documents. Specifically, we use t-SNE[10] [23] to generate the scatter plots for the document latent semantic vectors

[10]https://lvdmaaten.github.io/tsne/

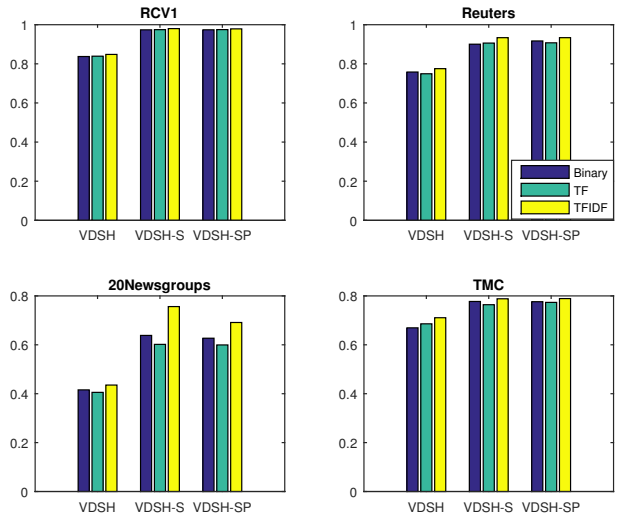



**Figure 3: Precision@100 for different term weighting schemes on the proposed models with the 32-bit hash code.**

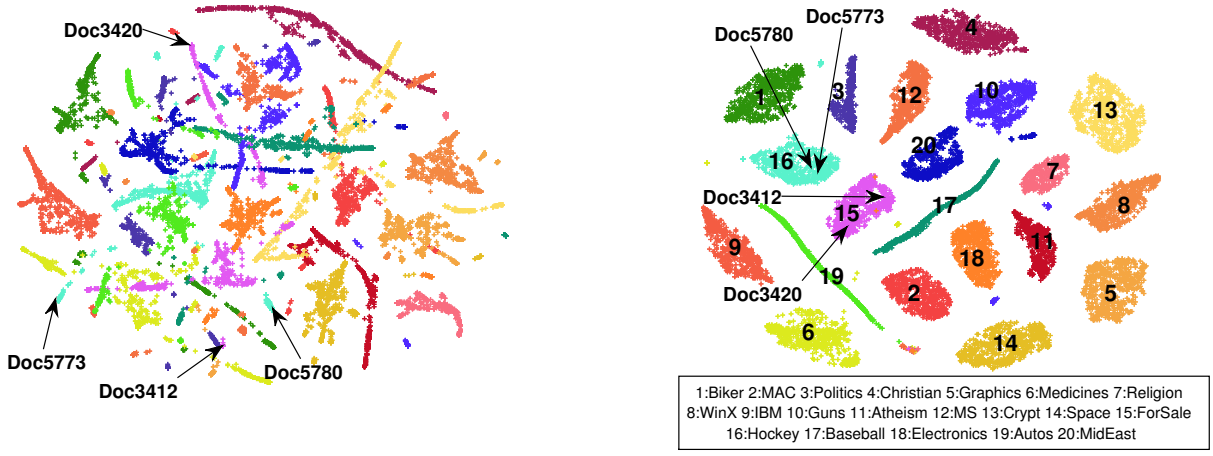
in 32-dimensional space obtained by SHTTM and VDSH-S on the 20Newsgroup dataset. Figure 4 shows the results. Here, each data point represents a document which is associated with one of the 20 categories. Different colors represent different categories based on the ground truth.

As we can see in Figure (4)(b), VDSH-S generates well separated clusters with each corresponding to a true category (each number in the plot represents a category ID). On the other hand, the clustering structure from SHTTM shown in Figure (4)(a) is much less evident and recognizable. Some closeby clusters in Figure (4)(b) are also semantically related, e.g., Category 7 (Religion) and Category 11 (Atheism); Category 20 (Middle East) and Category 10 (Guns); Category 8 (WinX) and Category 5 (Graphics).

We further sampled some documents from the dataset and see where they are represented in the plots. Table 3 contains the DocIDs, categories, and subjects of the sample documents. Doc5780 discusses some trade rumor in NHL and Doc5773 is about NHL team leaders. Both documents belong to the category of Hockey and should be close to each other, which can be clearly observed in Figure (4)(b) by VDSH-S. However, these two documents are projected far away from each other by SHTTM as shown in Figure (4)(a). For another random pair of documents Doc3420 and Doc3412 in the plots, VDSH-S also puts them much closer to each other than SHTTM does. These results demonstrate the great effectiveness of VDSH-S in learning low-dimensional representations for text documents.

# 6 CONCLUSIONS AND FUTURE WORK

Text hashing has become an important component in many large-scale information retrieval systems. It attempts to map documents in a high-dimensional space into a low-dimensional compact representation, while preserving the semantic relationship of the documents as much as possible. Deep learning is a powerful representation learning approach and has demonstrated its effectiveness of




(a) SHTTM (b) VDSH-S

**Figure 4: Visualization of the 32-dimensional document latent semantic vectors by SHTTM and VDSH-S on the 20Newsgroup dataset using t-SNE. Each point represents a document and different colors denote different categories based on the ground truth. In (b)VDSH-S, each number is a category ID and the corresponding categories are shown below the plot.**

| DocId | Category | Title/Subject |
|---|---|---|
| Doc5780 | Hockey | Trade rumor: Montreal/Ottawa/Phillie |
| Doc5773 | Hockey | NHL team leaders in +/- |
| Doc3420 | ForSale | Books For Sale [Ann Arbor, MI] |
| Doc3412 | ForSale | *** NeXTstation 8/105 For Sale *** |

**Table 3: The titles of the four sample documents in Figure 4**

learning effective representations in a wide range of applications, but there is very little prior work on utilizing it for text hashing tasks. In this paper, we exploit the recent advances in variational autoencoder and propose a series of deep generative models for text hashing. The models enjoy the advantages of both deep learning and probabilistic generative models. They can learn subtle nonlinear semantic representation in a principled probabilistic framework, especially when supervisory signals are available. The experimental results on four public testbeds demonstrate that the proposed supervised models significantly outperform the competitive baselines.

This work is an initial step towards a promising research direction. The probabilistic formulation and deep learning architecture provide a flexible framework for model extensions. In future work, we will explore deeper and more sophisticated architectures such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) [19], autoregressive neural network (NADE, MADE) [5, 18] for encoder and decoder. These more sophisticated models may be able to capture the local relations (by CNN) or sequential information (by RNN, NADE, MADE) in text. Moreover, we will utilize the probabilistic generative process to sample and simulate new text, which may facilitate the task of Natural Language Generation [27]. Last but not least, we will adapt the proposed models to hash other types of data such as images and videos.

## APPENDIX

In this section, we show the derivations of the proposed models. Due to the page limit, we only focus on VDSH-SP, the most sophisticated one among the three models. The other two models can be similarly derived.

The likelihood of document $d$ and labels $Y$ is:

$$\log P(d;Y) = \log \int_{s;v} P(d;Y;s;v)\,ds\,dv$$
$$= \log \int_{s;v} Q(s;v|d;Y)\frac{P(d;Y;s;v)}{Q(s;v|d;Y)}\,ds\,dv$$

Based on the Jensen's Inequality [8],

$$\log P(d;Y) \quad E_{Q(s;v)}[\log P(d;Y;s;v) \quad \log Q(s;v|d;Y)]$$
$$= E_{Q(s;v)}[\log P(d|s;v)P(Y|s)] + E_{Q(s;v)}[\log P(s)$$
$$+ \log P(v) \quad \log Q(s;v|d;Y)] \tag{6}$$
$$= E_{Q(s;v)}[\log P(d|s;v)P(Y|s)] + E_{Q(s;v)}[\log P(s) \quad \log Q(s|d)]$$
$$+ E_{Q(s;v)}[\log P(v) \quad \log Q(v|d)] \tag{7}$$
$$= E_{Q(s;v)}[\log P(d|s;v)P(Y|s)] \quad D_{KL}(Q(s|d) \,\|\, P(s))$$
$$D_{KL}(Q(v|d) \,\|\, P(v)) \tag{8}$$
$$= E_{Q(s;v)}[\log P(d|s;v)] + E_{Q(s;v)}[\log P(Y|s)]$$
$$D_{KL}(Q(s|d) \,\|\, P(s)) \quad D_{KL}(Q(v|d) \,\|\, P(v)) \tag{9}$$
$$= E_{Q(s;v)}\Big[\sum_{i=1}^{|d|}\log P(w_i|f(s+v;\theta)) + \sum_{j=1}^{|Y|}\log P(y_j|f(s;\tau))\Big]$$
$$D_{KL}(Q(s|d;\phi) \,\|\, P(s)) \quad D_{KL}(Q(v|d;\phi) \,\|\, P(v)) \tag{10}$$

In Eqn.(6), we factorize the joint probability based on the generative process. Thus, $P(d;Y;s;v) = P(d|s;v)P(Y|s)P(s)P(v)$. In Eqn.(7), the variational distribution, $Q(s;v|d;Y)$ is equal to the product

of $Q(s|d)$ and $Q(v|d)$ by assuming the conditional independence of $s, v, Y$ given $d$. Eqn.(8) and Eqn.(9) are the results of rearranging and simplifying terms in Eqn.(7). Plugging the individual words and labels, we obtain the final lowerbound objective function in Eqn.(10) (also in Eqn.(5)).

Because of the Gaussian assumptions on latent semantic vector $s$ and latent private variable $v$, the two KL divergences in Eqn.(10) have analytic forms. We let $\mu_s$ and $\sigma_s$ are mean and standard deviation of $s$. $\mu_v$ and $\sigma_v$ are similar defined. We use subscript $k$ to denote the $k^{th}$ element of the vector. The following derivation is an analytical form for a single KL divergence term:

$$D_{KL}(Q(s|d;\phi) \,\|\, P(s)) = \mathbb{E}_{Q(s)}[\log P(s)] \quad \mathbb{E}_{Q(s)}[\log Q(s|d;\phi)]$$
$$= \frac{1}{2}\sum_{k=1}^{K}(1 + \log\sigma_{s,k}^2 \quad \mu_{s,k}^2 \quad \sigma_{s,k}^2) \tag{11}$$

$D_{KL}(Q(v|d;\phi) \,\|\, P(v))$ can be derived in the same way. The expectation terms in Eqn.(10) do not have a closed form solution, but we can approximate them by the Monte Carlo simulation as follows:

$$\mathbb{E}_{Q(s,v)}[\log P(d|f(s+v;\theta))] + \mathbb{E}_{Q(s,v)}[\log P(Y|s)]$$
$$\simeq \frac{1}{M}\sum_{m=1}^{M}\log P(d|f(s^{(m)}+v^{(m)};\theta)) + \log P(Y|s^{(m)}) \tag{12}$$

The superscript $m$ denotes the $m^{th}$ sample. By shift and scale transformation, we have $s^{(m)} = \epsilon_s^{(m)} \quad \sigma_s + \mu_s$. We denote $\epsilon_s$ as a sample drawn from a standard multivariate normal and $$ is an element-wise multiplication. Also, $v^{(m)}$ is obtained in the same way, $v^{(m)} = \epsilon_v^{(m)} \quad \sigma_v + \mu_v$. By using this trick, we can obtain multiple samples of $\epsilon$ and feed them as the deterministic input to the neural network. The model becomes an end-to-end deterministic deep neural network with the following objective function:

$$L \simeq \frac{1}{M}\sum_{m=1}^{M}\log P(d|f(s^{(m)}+v^{(m)};\theta)) + \log P(Y|s^{(m)})$$
$$+ \frac{1}{2}\sum_{k=1}^{K}(2 + \log\sigma_{s,k}^2 \quad \mu_{s,k}^2 \quad \sigma_{s,k}^2 + \log\sigma_{v,k}^2 \quad \mu_{v,k}^2 \quad \sigma_{v,k}^2) \tag{13}$$

## ACKNOWLEDGMENTS

The authors would like to thank Mengwen Liu, for her many helpful comments and suggestions on the first draft, Travis Ebesu for his drawing on the model architectures in Figure 1, and anonymous reviewers for valuable comments and feedback.

## REFERENCES

[1] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
[2] 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.
[3] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
[4] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.
[5] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *ICML*, pages 881–889, 2015.
[6] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.
[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.
[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.
[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.
[10] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, pages 55–64. ACM, 2016.
[11] G. E. Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.
[12] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM, 2013.
[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.
[14] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, pages 3581–3589, 2014.
[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.
[16] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.
[17] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.
[18] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2, 2011.
[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[20] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPR Workshops*, pages 27–35, 2015.
[21] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE, 2012.
[22] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.
[23] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.
[24] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge: Cambridge university press, 2008.
[25] Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *ICML*, 2016.
[26] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.
[27] E. Reiter, R. Dale, and Z. Feng. *Building natural language generation systems*, volume 33. Cambridge: Cambridge university Press, 2000.
[28] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, 2014.
[29] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.
[30] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, pages 373–382. ACM, 2015.
[31] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.
[32] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.
[33] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431. IEEE, 2010.
[34] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.
[35] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *ICCV*, pages 3032–3039, 2013.
[36] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *TPAMI*, 2017.
[37] Q. Wang, D. Zhang, and L. Si. Semantic hashing using tags and topic modeling. In *SIGIR*, pages 213–222. ACM, 2013.
[38] W. Wang, H. Lee, and K. Livescu. Deep variational canonical correlation analysis. *arXiv preprint arXiv:1610.03454*, 2016.
[39] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.
[40] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014.
[41] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, and H. Hao. Convolutional neural networks for text hashing. In *AAAI*, pages 1369–1375. AAAI Press, 2015.
[42] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25. ACM, 2010.