Deep Latent-Variable Models for Natural Language Understanding and Generation

by

Dinghan Shen

Department of Electrical and Computer Engineering
Duke University

Date: ______________________________

Approved:

____

Lawrence Carin, Advisor

____
Guillermo Sapiro

____
Yiran Chen

____
Rong Ge

____
Ricardo Henao Giraldo

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2020

<u>ABSTRACT</u>

Deep Latent-Variable Models for Natural Language Understanding and Generation

by

Dinghan Shen

Department of Electrical and Computer Engineering
Duke University

Date: ______________________
Approved:

____

Lawrence Carin, Advisor

____
Guillermo Sapiro

____
Yiran Chen

____
Rong Ge

____
Ricardo Henao Giraldo

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University

2020

Abstract

Deep latent-variable models have been widely adopted to model various types of data, due to its ability to: 1) infer rich high-level information from the input data (especially in a low-resource setting); 2) result in a generative network that can synthesize samples unseen during training. In this dissertation, I will present the contributions I have made to leverage the general framework of latent-variable model to various natural language processing problems, which is especially challenging given the discrete nature of text sequences. Specifically, the dissertation is divided into two parts.

In the first part, I will present two of my recent explorations on leveraging deep latent-variable models for natural language understanding. The goal here is to learn meaningful text representations that can be helpful for tasks such as sentence classification, natural language inference, question answering, *etc*. Firstly, I will propose a variational autoencoder based on textual data to digest unlabeled information. To alleviate the observed *posterior collapse* issue, a specially-designed deconvolutional decoder is employed as the generative network. The resulting sentence embeddings greatly boost the downstream tasks performances. Then I will present a model to learn compressed/binary sentence embeddings, which is storage-efficient and applicable to on-device applications.

As to the second part, I will introduce a multi-level Variational Autoencoder (VAE) to model long-form text sequences (with as many as 60 words). A multi-level generative network is leveraged to capture the word-level, sentence-level coherence, respectively. Moreover, with a hierarchical design of the latent space, long-form and coherent texts can be more reliably produced (relative to baseline text VAE models). Semantically-rich latent representations are also obtained in such an unsupervised manner. Human evaluation further demonstrates the superiority of the proposed method.

Acknowledgements

My Ph.D. study was an amazing journey, and I really enjoyed every moment of it. However, there were also ups and downs. I am lucky enough to get the support and help from many people along the way. In this section, I would like to take a moment to express my sincere gratitude to all of them.

Firstly, I wish to express my sincere appreciation to my advisor, Professor Lawrence Carin, who let me explore different research topics I was interested in with incredible open-mindedness. Larry is a great leader, and he has taught me how to conduct research in the right way, gave me opportunities to improve my different skill sets, and pushed me to become the best of myself. Thanks for coaching, supporting and motivating me along this journey!

Besides, I would like to show my deepest gratitude to my dissertation committee members, Professors Guillermo Sapiro, Yiran Chen, Rong Ge, Ricardo Henao Giraldo for their time and valuable feedback. I also want to thank Professors Galen Reeves and Robert Calderbank for serving as the committee members on my qualifying exam.

I am also very grateful to my mentors/hosts of the internships during my Ph.D. study: Martin Renqiang Min at Machine Learning Group, NEC Labs America, Asli Celikyilmaz and Jianfeng Gao at MSR AI, Microsoft Research, and Dipanjan Das and Ankur Parikh at Language Team, Google AI. Thanks to their support and guidance, these intern experiences in industry really broaden my horizons.

Moreover, I wanted to thank all my fellow group members. I was lucky to collaborate with many of them and learn a lot through the process. It includes: Xuejun Liao, Ricardo Henao, , Shaobo Han, David Carlson, Changwei Hu, Kyle Ulrich, Yizhe Zhang, Chunyuan Li, Zhe Gan, Yunchen Pu, Kai Fan, Changyou Chen, Andrew Stevens, Qinliang Su, Jinghao Lu, Zhao Song, Xinyuan Zhang, Yitong Li, Wenlin Wang, Kevin Liang, Gregory Spell, Guoyin Wang, Liqun Chen, Shuyang Dai, Jianqiao Li, Paidamoyo Chapfuwa, Ruiyi Zhang, Chenyang Tao, Hongteng Xu, Dong Wang, Johnnyy Sigman, Hao Fu, Pengyu Cheng, Ke Bai, Siyang Yuan, Yan Zhao, Weiyao Wang, and Hao Liu, Wei Tuo Hao, Jiachang Liu, Junya Chen, Qian Yang, Lei Zhang, Yulai Cong, Shijing Si, Dhanasekar Sundararaman, Nikhil Mehta, Vivek Subramanian, Shounak Dutta.

Lastly but most importantly, I cannot express enough gratitude to my parents for their unconditional love, the greatest thing I am blessed to have. Unfortunately, my father passed away a few months ago, but I believe that he can still see what I am doing and is proud of it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing is a core to many AI systems, such as machine translation, virtual assistant, document understanding platform, *etc*. With deep learning becoming the current dominant paradigm for natural language processing, this field has attracted much attention in recent years. Some major successes include machine translation, language modeling, natural language inference, question answering, *etc*, where models based on deep neural networks have consistently demonstrated state-of-the-art empirical results.

1.1 Latent-variable models for text processing

Sequence-to-sequence models [SVL14] are the most common strategy for obtaining robust sentence representations, as these are capable of leveraging information from unlabeled data. These models first encode the input sentence x (composed of T words, $w_{1:T}$) into a fixed-length vector $z = g(x)$, and then reconstruct/generate the output sequence from z. Specifically, in the autoencoder setup, the output of the decoder is the reconstruction of the input sentence x, denoted $\hat{x}$ with words $\hat{w}_{1:T}$,

$$\begin{aligned} p(\hat{x}|x) &= p(\hat{w}_{1:T}|w_{1:N}) \\ &= p(\hat{w}_1|z = g(x)) \prod_{t=2}^{T} p(\hat{w}_t|z = g(x), \hat{w}_{1:t-1}) \,, \end{aligned} \tag{1.1}$$

where $g(\cdot)$ is a *deterministic*, generally nonlinear transformation of x. The deterministic $g(x)$ may result in poor model generalization, especially when only a limited number of labeled data are available for training. Below we consider a *probabilistic* representation for z, *i.e.*, $p(z|x)$.

Recently [MYB16a] introduced a Neural Variational Inference (NVI) framework for text modeling, in which they infer a stochastic latent variable $z \sim q(z|x)$ to model the input text, constructing an inference network to approximate the true posterior distribution $p(z|x)$. This strategy endows latent variable z with a better ability to generalize [MYB16a]. Conditioning on the latent code z, a decoder network $p(x|z)$ maps z back to reconstruct the original sequence, x. Given a set of observed sentences (training set), the parameters of this model are learned by maximizing the marginal $p(x)$. Since this is intractable in most cases, a variational lower bound is typically employed as the objective to be maximized [KW13]:

$$\begin{aligned}\mathcal{L}_{\text{vae}} &= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)|p(z)) \\ &= E_{q_\phi(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)] \\ &\leq \log \int p_\theta(x|z)p(z)dz = \log p_\theta(x)\,, \end{aligned} \tag{1.2}$$

where θ and ϕ denote decoder and encoder parameters, respectively. The lower bound $\mathcal{L}_{\text{vae}}(\theta, \phi; x)$ is maximized w.r.t. both encoder and decoder parameters. Intuitively, the model aims to minimize the reconstruction error as well as to regularize the posterior distribution $q_\phi(z|x)$ as to not diverge too much from the prior $p(z)$. This neural variational inference framework has achieved significant success on other types of data, such as images [GDG+15, PGH+16a].

1.2 Challenges with the NVI framework for text

Extracting sentence features for text with the above NVI framework has been shown to be difficult [BVV+16a, YHSBK17a]. For an unsupervised latent-variable model, which is often referred to as a variational autoencoder [KW13], the parameters are optimized by minimizing the reconstruction error of sentences, as well as regularizing the posterior distribution $q_\phi(z|x)$ to be close to the prior $p(z)$, as in (4.1) via $D_{KL}(q_\phi(z|x)|p(z))$. There-

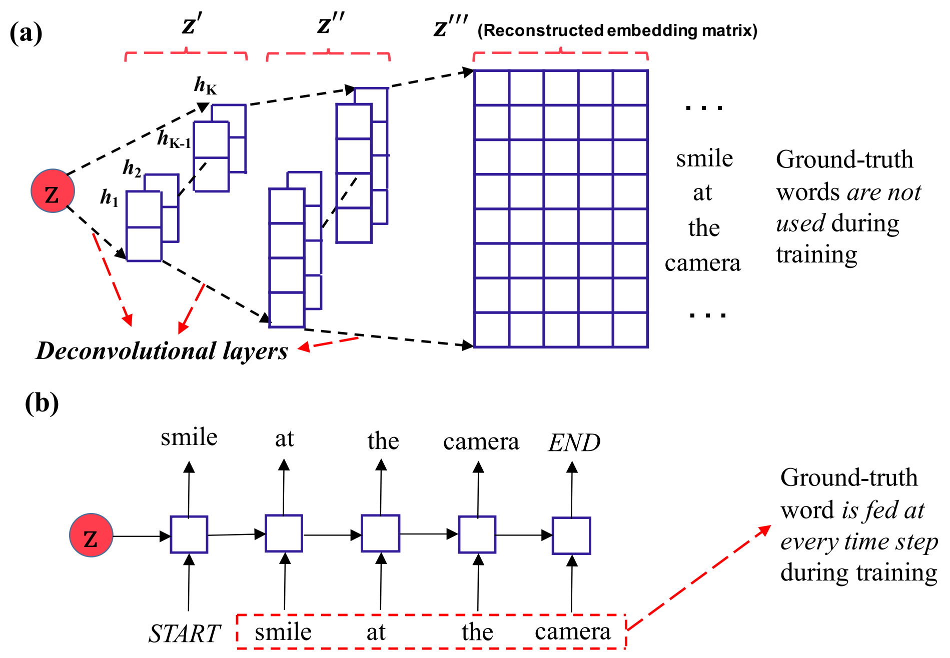
(a)
z'
z''
z''' (Reconstructed embedding matrix)
h_K
h_K-1
h_2
h_1
Z
...
smile
at
the
camera
...
Ground-truth words are not used during training
Deconvolutional layers
(b)
smile
at
the
camera
END
Z
START
smile
at
the
camera
Ground-truth word is fed at every time step during training

Figure 1.1: (a) Diagram of deconvolutional sequence decoder, comparing with (b) LSTM sequence decoder.

fore, we can think of the variational autoencoder as a regularized version of a standard (deterministic) autoencoder (sequence-to-sequence model), due to the additional penalty term coming from KL divergence loss.

Although the KL divergence in (4.1) term plays a key role in training latent-variable models with the NVI framework, it has been reported that, when applied to text data (sentences), the KL loss tends to be insignificantly small during training [BVV+16a]. As a result, the encoder matches the Gaussian prior regardless of the input, and the decoder doesn't take advantage of information from the latent variable z. Moreover, it has been reported that poor results in this setting may be attributed to the autoregressive nature of the LSTM decoder [CKS+16, BVV+16a]. While decoding, the LSTM imposes strong conditional dependencies between consecutive words, thus, from (1.1), the information from z becomes less impactful during learning. Motivated by these issues, [YHSBK17a] employed dilated CNNs, instead of the LSTM, as a sentence decoder for a latent-variable model. In [YHSBK17a] the latent variable z is able to encode more semantic information, because of the smaller contextual capacity of the dilated CNN decoder. However, optimization challenges remain, because ground-truth words are employed while training, as the dilated CNN is an autoregressive decoder. Consequently, the inferred latent codes cannot be considered as global features of a sentence, since they do not necessarily encode all the information needed to reconstruct an entire sequence.

1.3 VAE for text generation

VAEs trained under the neural variational inference (NVI) framework, has been widely used for generating text sequences: [BVV+16b, YHSBK17b, SSB17b, MYB16a, SSL+17a, MGB17a, ZZE17, SSL+17e, GHOL18, KWM+18, YZHN18, KRV+18, BMVP18, CTLG18, DKC+18, SB18].

By encouraging the latent feature space to match a prior distribution within an encoder-decoder architecture, the learned latent variable could potentially encode high-level semantic features and serve as a global representation during the decoding process [BVV+16b]. The generated results are also endowed with better diversity due to the sampling procedure of the latent codes [ZZE17]. Generative Adversarial Networks (GANs) [YZWY17, HYL+17, ZGF+17a, FGD18, CDT+18], is another type of generative models that are commonly used for text generation. However, existing works have mostly focused on generating one sentence (or multiple sentences with at most twenty words in total). The task of generating relatively longer units of text has been less explored.

1.4 VAE for learning discrete representations

Models with discrete random variables have attracted much attention in the deep learning community [JGP16, MMT16, vdOV+17, LSHT17, SN17]. Some of these structures are more natural choices for language or speech data, which are inherently discrete. More specifically, [vdOV^{+}17] combined VAEs with vector quantization to learn discrete latent representation, and demonstrated the utility of these learned representations on images, videos, and speech data. [LSHT17] leveraged both pairwise label and classification information to learn discrete hash codes, which exhibit state-of-the-art performance on image retrieval tasks.

For natural language processing (NLP), although significant research has been made to learn *continuous* deep representations for words or documents [MSC^{+}13, KZS^{+}15b, SWW^{+}18], *discrete* neural representations have been mainly explored in learning word embeddings [SN17]. In these recent works, words are represented as a vector of discrete numbers, which are very efficient storage-wise, while showing comparable performance on several NLP tasks, relative to continuous word embeddings. However, discrete representations that are learned in an *end-to-end* manner at the *sentence* or *document* level have been rarely explored. Also there is a lack of strict evaluation regarding their ness. Our work focuses on learning discrete (binary) representations for text documents. Further, we employ semantic hashing (fast similarity search) as a mechanism to evaluate the quality of learned binary latent codes.

Inspired by these observations, this dissertation makes several contributions to the aforementioned issues. Specifically, Section 2 focuses on how to design a latent-variable model for semi-supervised learning, where unlabeled needs to be effectively exploited with limited number of labeled samples. Natural language inference and paraphrase identification has been chosen to evaluate the proposed methods. Section 3 proposes to learn binarized sentence embeddings with deep latent-variable models. This type of representations is storage-efficient, and thus can be employed for on-device NLP applications. It is demonstrated experimentally that our model make effective use of unlabeled data, and greatly improve over strong baseline models. In Section 4, we introduce a multi-level variational autoencoder for long-form text generation. The model not only contains a hierarchical decoder (where both word-level and sentence-level coherence can be captured), but also assume a hierarchical latent space, and thus endow the learned representations with more flexibility. Finally, in Section 5, I show that the idea of learning binarized sentence representations can be extended to a more general scenario, where a pre-trained sentence encoder can be utilized for a wide variety of downstream NLP tasks.

Chapter 2

Deconvolutional Latent-Variable Model for Text Sequence Matching

In this chapter, I will introduce a latent-variable model for text matching, inferring sentence representations by jointly optimizing generative and discriminative objectives. To alleviate typical optimization challenges in latent-variable models for text, we employ deconvolutional networks as the sequence decoder (generator), providing learned latent codes with more semantic information and better generalization. Our model, trained in an unsupervised manner, yields stronger empirical predictive performance than a decoder based on Long Short-Term Memory (LSTM), with less parameters and considerably faster training. Further, we apply it to text sequence-matching problems. The proposed model significantly outperforms several strong sentence-encoding baselines, especially in the semi-supervised setting.

2.1 Introduction

The ability to infer the degree of match between two text sequences, and determine their semantic relationship, is of central importance in natural language understanding and reasoning [BGWB14]. With recent advances in deep neural networks, considerable research has focused on developing *end-to-end* deep learning models for text sequence matching [HLLC14a, WJ16, RGH$^+$15, WHF17, SMLC17a]. State-of-the-art models typically first encode the text sequences into hidden units via a Long Short term Memory (LSTM) model or a Convolutional Neural Network (CNN), and techniques like attention mechanisms [RGH$^+$15] or memory networks [HBCW15] are subsequently applied for the final sequence matching, usually addressed as a classification problem. However, the word-by-word matching nature of these models typically gives rise to high computational complexity, either $\mathcal{O}(T^2)$ [WJ16] or $\mathcal{O}(T)$ [RGH+15], where T is the sentence length. Therefore, these approaches are computationally expensive and difficult to scale to large datasets or long text sequences.

Another class of models for matching natural language sentences is based on *sentence encoding* methods, where each sentence is mapped to a vector (embedding), and two such vectors are used for predictions of relationships between the corresponding two sentences [BGR+16, MML+15]. In this case the matching complexity is independent of sentence length. However, it has been found that is hard to encode the semantic information of an entire sequence into a single vector [BAPM15b].

For these models, it is important to learn an informative sentence representation with two properties: (*i*) it preserves its fundamental details, *e.g.*, n-gram fragments within the sequence of text; (*ii*) the learned representation should contain discriminative information regarding its relationship with the target sequence. So motivated, we propose to infer the embedding for each sentence with *deep generative models*, due to their ability to make effective use of unlabeled data and learn abstract features from complex data [KMRW14a, YHSBK17a, PGH+16a, WPV+18]. Moreover, the objective of a generative model addresses generation/reconstruction, and thus learns latent codes that naturally preserve essential information of a sequence, making them particularly well suited to sentence matching.

Recent advances in neural variational inference have manifested deep latent-variable models for text [MYB16a]. The general idea is to map the sentence into a continuous latent variable, or *code*, via an inference network (encoder), and then use the generative network (decoder) to reconstruct the input sentence conditioned on samples from the latent code (via its posterior distribution). As a first attempt, [BVV+16a] proposed a Variational Auto-Encoder (VAE)-based generative model for text, with LSTM networks [HS97] as the sequence decoder. However, due to the recurrent nature of the LSTM decoder, the model tends to largely ignore information from the latent variable; the learned sentence embedding contains little information from the input, even with several training modifications [BVV$^+$16a]. To mitigate this issue, [YHSBK17a] proposed to use a dilated CNN, rather than an LSTM, as a decoder in their latent-variable model. Since this decoder is less dependent on the contextual information from previous words, the latent-variable representation tends to encode more information from the input sequence.

Unfortunately, regardless of whether LSTMs or dilated CNNs are used as the generative network, ground-truth words need to be fed into the decoder during training, which has two potential issues: (*i*) given the powerful recursive and autoregressive nature of these decoders, the latent-variable model tends to ignore the latent vector altogether, thus reducing to a *pure* language model (without external inputs) *i.e.*, latent representations are not effective during training [BVV$^+$16a, CKS$^+$16]; (*ii*) the learned latent vector does not necessarily encode all the information needed to reconstruct the entire sequence, since additional guidance is provided while generating every word, *i.e.*, *exposure bias* [RCAZ15].

We propose *deconvolutional networks* as the sequence decoder in a latent-variable model, for matching natural language sentences. Without any recurrent structure in the decoder, the typical optimization issues associated with training latent-variable models for text are mitigated. Further, global sentence representations can be effectively learned, since no ground-truth words are made available to the decoder during training.

In the experiments, we first evaluate our deconvolution-based model in an unsupervised manner, and examine whether the learned embedding can automatically distinguish different writing styles. We demonstrate that the latent codes from our model are more informative than LSTM-based models, while achieving higher classification accuracy. We then apply our latent-variable model to text-sequence matching tasks, where predictions are made only based on samples from the latent variables. Consequently, without any prior knowledge on language structure, such as that used in traditional text analysis approaches (*e.g.*, via a parse tree), our deconvolutional latent-variable model outperforms several competitive baselines, especially in the semi-supervised setting.

Our main contributions are as follows:

i) We propose a neural variational inference framework for matching natural language sentences, which effectively leverages unlabeled data and achieves promising results with little supervision.

ii) We employ deconvolutional networks as the sequence decoder, alleviating the optimization difficulties of training latent-variable models for text, resulting in more informative latent sentence representations.

iii) The proposed deconvolutional latent-variable model is highly parallelizable, with less parameters and much faster training than LSTM-based alternatives.

2.2 Background

2.2.1 Matching natural language sentences

Assume we have two sentences for which we wish to compute the degree of match. For notational simplicity, we describe our model in the context of Recognizing Textual Entailment (RTE) [RGH$^+$15], thus we denote the two sequences as *P* for premise and *H* for hypothesis, where each sentence pair can be represented as (p_i, h_i), for $i = 1, 2, 3..., N$, where N is the total number of pairs. The goal of sequence matching is to predict judgement y_i for the corresponding sentence pair, by modeling the conditional distribution $p(y_i|p_i, h_i)$, where $y_i \in \{$*entailment*, *contradiction*, *neutral*$\}$. *Entailment* indicates that p_i and h_i can be inferred from each other, *contradiction* suggests they have opposite semantic meanings, while *neutral* means p_i and h_i are irrelevant to each other. This framework can be generalized to other natural language processing applications, such as paraphrase identification, where $y_i = 1$ if p_i is a paraphrase of h_i, and $y_i = 0$ otherwise. In this regard, text sequence matching can be viewed as either a binary or multi-class classification problem [YHBP14].

Although word/phrase-level attention [RGH$^+$15] or matching strategies [WJ16] are often applied to text sequence-matching problems, we only consider *sentence encoding-based models*, because of their promising low complexity. Specifically, our model is based on the *siamese* architecture [BGL$^+$94], which consists of a twin network that processes natural language sentence pairs independently (the parameters of the twin network are tied); there is no interaction before both sentence representations are inferred. A classification layer is built on top of the two latent representations, for final prediction (matching).

The shared encoder network can be designed as any form of nonlinear transformation, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) or Multi-Layer Perceptrons (MLPs). However, to effectively match natural language sentences with the *siamese* architecture, the key is to learn informative sentence representations through the encoder network. To this end, below we describe use of CNNs in the context of a latent-variable model.

2.3 Model

2.3.1 Deconvolutional sequence decoder

Deconvolutional networks, also known as *transposed* convolutional layers, are typically used in deep learning models to up-sample fixed-length latent representations or high-level feature maps [ZKTF10]. Although widely adopted in image generative models, deconvolutional networks have been rarely applied to generative models for text. To understand the form of the decoder needed for text, we first consider the associated convolutional encoder ([Kim14], [ZSW$^+$17]). The text is represented as a matrix, with "width" dictated by the sentence length and "height" dictated by the dimensionality of the word embeddings. With K_1 convolutional filters at layer 1 of the model, after one-dimensional (1D) convolution between the 2D filters and 2D sentence embedding matrix (convolution in the direction of the word index, or "time"), K_1 1D signals are manifested. Using these K_1 1D feature maps, a similar process repeats to substantiate subsequent layers in the deep model. Hence, at layer l of the model, there are K_l 1D signals manifested from K_l 1D convolutions between K_l 2D filters and the 2D feature-map from layer $l-1$.

The encoder discussed above starts at the "bottom" with the sentence-embedding matrix, and works upward to the latent code z. The decoder works downward, starting at z and arriving at the sentence-embedding matrix. Specifically, the decoder network takes as input $z \in \mathbb{R}^M$ sampled from the inference (encoder) network $q_\phi(z|x)$. For an L-layer decoder model, the feature maps at layer L (just beneath the latent code z) are manifested by K_L filter matrices $f_i^{(L)} \in \mathbb{R}^{H_L \times M}$, for $i = 1, 2, ..., K_L$, where H_L corresponds to the number of components in the temporal (word) dimension. Each 2D matrix $f_i^{(L)}$ is multiplied by column vector z (transpose convolution), yielding K_L 1D feature maps. This yields an $H_L \times K_L$ feature-map matrix at layer L (followed by ReLU pointwise nonlinearity). To yield the layer $L-1$ feature map matrix, the process repeats, using filters $f_i^{(L-1)} \in \mathbb{R}^{H_{L-1} \times K_L}$, for $i = 1, 2, ..., K_{L-1}$, with which K_{L-1} 1D convolutions are performed with the feature-map matrix from layer L (convolutions in the temporal/word dimension). This again yields a feature-map matrix at layer $L-1$, followed by ReLU nonlinearity.

This process continues sequentially, until we arrive at the bottom of the decoder network, yielding a final matrix from which the sentence-embedding matrix is approximated. To be explicit, in Fig. 1.1 let z' and z'' represent the feature-map matrices at the top-two layers of a three-layer model. Let z''' represent the matrix recovered at the bottom layer of the network through the above process, with "height" corresponding to the dimension of the word-embedding. Suppose $\boldsymbol{E}$ is the word-embedding matrix for our vocabulary, and $\hat{w}_i$ the ith word in the reconstructed sentence. We compute the probability that $\hat{w}_i$ is word

s as:

$$p(\hat{w}_i = s) = \frac{\exp\{\tau^{-1}\cos(z_i''', \boldsymbol{E}[s])\}}{\sum_{s' \in V}\exp\{\tau^{-1}\cos(z_i''', \boldsymbol{E}[s'])\}}, \tag{2.1}$$

where $\cos(a, b)$ is the *cosine similarity* between vectors a and b, V is the vocabulary which contains all possible words and $\boldsymbol{E}[s]$ represents the column of $\boldsymbol{E}$ corresponding to word s; z_i''' is the i-th column of the up-sampled representation z'''. Parameter τ controls the sparsity of resulting probabilities, which we denote as the *temperature* parameter. We set $\tau = 0.01$ in our experiments.

The multilayer coarse-to-fine process (latent variable vector to embedding matrix) implied by repeatedly applying the above decoder process illustrated in Figure 1.1(a)) has two advantages: i) it reflects the natural hierarchical tree structure of sentences, thus may better represent syntactic features, which is useful when reconstructing sentences; ii) the deconvolutional network allows for efficient parallelization while generating each fragment of a sentence, and thus can be considerably faster than an LSTM decoder.

As shown in Figure 1.1, the training procedures for deconvolutional (a) and LSTM (b) decoders are intrinsically different. In the latter, ground-truth words of the previous time steps are provided while training the network. In contrast, the deconvolutional network generates the entire sentence (in block) from z alone. Because of this distinction, the LSTM decoder, as an autoregressive model with powerful recurrence, tends to explain all structure in the data, with little insight from the latent variables which only provide information at the beginning of the sentence, thus acting merely as a prior.

2.3.2 Deconvolutional latent-variable models

In this section we incorporate the deconvolutional sequence decoder described in the previous section in our latent-variable model for text. Because of the coarse-to-fine generation process described above, the model does not have partial access to observed data (ground-truth words) during the generation process, as in an LSTM, thus the latent-variable model must learn to encode as much information as possible from the input alone. Moreover, in this way the learned latent code can be truly viewed as a global feature representation of sentences, since it contains all the essential information to generate the text sequence. In the following, we describe the proposed deconvolutional latent-variable models, in the context of both unsupervised and supervised (including semi-supervised) learning.

Unsupervised sequence learning

To demonstrate the effectiveness of our proposed model, we explore training it in an unsupervised manner. Specifically, for a input sentence x, the latent code is inferred through an encoder network $q_\phi(z|x)$ implemented as

$$\begin{aligned} \mu &= g_1(f^{\text{cnn}}(x;\phi_{10});\phi_{11}), & \log\sigma &= g_2(f^{\text{cnn}}(x;\phi_{20});\phi_{21}) \\ \varepsilon &\sim \mathcal{N}(0,\mathbf{I}), & z &= \mu + \varepsilon \odot \sigma\,, \end{aligned} \tag{2.2}$$

where $f^{\text{cnn}}(x;\phi_{10})$ denotes the transformation function of the encoder, accomplished via learning a CNN with input x and parameters ϕ_{10}, and $\odot$ represents the Hadamard vector product. The posterior mean μ and variance σ are generated through two non-linear transformations $g_1(\cdot)$ and $g_2(\cdot)$, both parameterized as neural networks; $g_1(y;\phi_{11})$ has input y and parameters ϕ_{11}. Note that (2.2) is $q_\phi(z|x)$ in (4.1), where $\phi = \{\phi_{10},\phi_{11},\phi_{20},\phi_{21}\}$. Then z is sampled with the re-parameterization trick [KW13] to facilitate model training. The sampled z is then fed into a deconvolutional sequence decoder described above, to reconstruct the corresponding input sentences. The model is trained by optimizing the variational lower bound in (4.1), without any discriminative information.

Supervised sequence matching

We apply our latent-variable model to text sequence-matching problems, employing the discriminative information encoded in latent code z (see Figure 2.1). For a sentence pair (p_i, h_i), the latent code for each sequence is inferred as in (2.2), where the parameters of

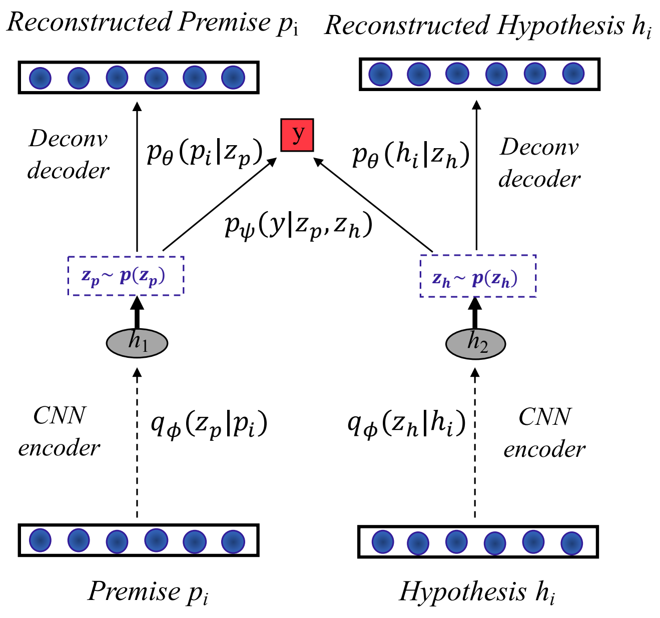

Figure 2.1: Our deconvolutional latent-variable model for text sequence matching.

the encoder network for z_p and z_h, premise and hypothesis, respectively, are shared. They are decoded by two shared-weight deconvolution networks, to recover the corresponding input sentence.

To infer the label, y, the two latent features are again sampled from the inference network and processed by a matching layer, to combine the information in the two sentences. This matching layer, defined as *heuristic* matching layer by [MML+15], can be specified as:

$$m = [z_p; z_h; z_p - z_h; z_p \odot z_h] \,,$$

These matching features are stacked together into $m \in \mathbb{R}^{4M}$, for $z_p, z_h \in \mathbb{R}^M$, and fed into a classifier. The classifier is a two-layer MLP followed by a fully-connected softmax layer, that outputs the probabilities for each label (entailment, contradiction and neutral), to model the conditional distribution $p_\psi(y|z_p, z_h)$, with parameters ψ.

To allow the model to explore and balance between maximizing the variational lower bound and minimizing the sequence matching loss, a joint training objective is employed:

$$\begin{aligned}\mathcal{L}^{\text{label}} = & -\mathcal{L}_{\text{vae}}(\theta, \phi; p_i) - \mathcal{L}_{\text{vae}}(\theta, \phi; h_i) \\ & + \alpha \mathcal{L}_{\text{match}}(\psi; z_p, z_h, y)\,,\end{aligned}$$

where ψ refers to parameters of the MLP classifier and α controls the relative weight between the generative loss, $\mathcal{L}_{\text{vae}}(\cdot)$, and sequence matching loss, $\mathcal{L}_{\text{match}}(\cdot)$, defined as the cross-entropy loss. When implementing this model, we anneal the value of α during training from 0 to 1 (the annealing rate is treated as a hyperparameter), so that the latent variable learned can gradually focus less on the reconstruction objective, only retaining those features that are useful for sequence matching, *i.e.*, minimizing the second term.

Extension to semi-supervised learning

Our latent-variable model can be readily extended to a semi-supervised scenario, where only a subset of sequence pairs have corresponding class labels. Suppose the empirical distributions for the labeled and unlabeled data are referred to as $\tilde{p}_l(P, H, y)$ and $\tilde{p}_u(P, H)$, respectively. The loss function for unlabeled data can be expressed as:

$$\mathcal{L}^{\text{unlabel}} = -\mathcal{L}_{\text{vae}}(\theta, \phi; p_i) - \mathcal{L}_{\text{vae}}(\theta, \phi; h_i)\,.$$

Therefore, the overall objective for the joint latent-variable model is:

$$\begin{aligned}\mathcal{L}_{\text{joint}} = & \,\mathbb{E}_{(p_i, h_i, y) \sim \tilde{p}_l}[\mathcal{L}^{\text{label}}(p_i, h_i, y)] \\ & + \mathbb{E}_{(p_i, h_i) \sim \tilde{p}_u}[\mathcal{L}^{\text{unlabel}}(p_i, h_i)]\,.\end{aligned} \tag{2.3}$$

To minimize $\mathcal{L}_{\text{joint}}$ w.r.t. θ, ϕ and ψ, we employ Monte Carlo integration to approximate the expectations in (2.3). In this case unlabeled data are leveraged in the objective via the standard VAE lower bound. During training, all parameters are jointly updated with stochastic gradient descent (SGD).

2.4 Experiments

2.4.1 Experimental Setup

Our deconvolutional latent-variable model can be trained in an unsupervised, supervised or semi-supervised manner. In this section we first train the model in an unsupervised way, with a mixed corpus of scientific and informal writing styles, and evaluate the sentence embeddings by checking whether they can automatically distinguish different sentence characteristics, *i.e.*, writing styles. Further, we apply our models to two standard text sequence matching tasks: Recognizing Textual Entailment (RTE) and paraphrase identification, in a semi-supervised setting. The summary statistics of both datasets are presented in Table 4.10.

For simplicity, we denote our deconvolutional latent-variable model as DeConv-LVM in all experiments. To facilitate comparison with prior work, several baseline models are implemented: (*i*) a basic Siamese model with CNNs as the encoder for both sentences, with sharing configurations and weights; (*ii*) an auto-encoder with CNN as the sequence encoder and DeConv as decoder; 3) a latent-variable model using a CNN as the inference network, and the generative network is implemented as an LSTM (denoted LSTM-LVM).

We use 3-layer convolutional neural networks for the inference/encoder network, in order to extract hierarchical representation of sentences ([HLLC14a]). Specifically, for all layers we set the filter window size (W) as 5, with a stride of 2. The feature maps (K) are set as 300, 600, 500, for layers 1 through 3, respectively. In our latent-variable models, the 500-dimension feature vector is then fed into two MLPs to infer the mean and variance of the latent variable z. The generative/decoder network is implemented as 3-layer deconvolutional networks, to decode the samples from latent variable z of size $M = 500$.

The model is trained using Adam [KB14a] with a learning rate of 3×10^{-4} for all parameters. Dropout [SHK$^+$14a] is employed on both word embedding and latent variable

Dataset	Train	Test	Classes	Vocabulary
Quora	384348	10000	2	10k
SNLI	549367	9824	3	20k

Table 2.1: Summary of text sequence matching datasets.

layers, with rates selected from $\{0.3, 0.5, 0.8\}$ on the validation set. We set the mini-batch size to 32. In semi-supervised sequence matching experiments, L_2 norm of the weight vectors is employed as a regularization term in the loss function, and the coefficient of the L_2 loss is treated as a hyperparameter and tuned on the validation set. All experiments are implemented in Tensorflow [AAB$^+$16], using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

2.4.2 Unsupervised Sentence Embedding

To investigate the effectiveness of our latent-variable model, we first train it in an unsupervised manner, using the dataset in [ZGF$^+$17b], where sentences from two corpora, *i.e*, *BookCorpus* dataset [ZKZ$^+$15] and the *arXiv* dataset, are merged together in equal proportion. The motivation here is to check whether the latent codes learned in our model can automatically distinguish between different writing styles, *i.e.*, sentences with scientific or informal styles represented by *BookCorpus* and *arXiv* dataset, respectively. In this experiment, our model is trained by optimizing the variational lower bound in (4.1), without any label/discriminative information provided. We compare our model with another latent-variable model using LSTM as the decoder, to especially highlight the contribution of the deconvolutional network to the overall setup. To ensure a fair comparison, we employ the same model architecture for the LSTM-based latent-variable model (LSTM-LVM), except for the decoder utilized. The LSTM hidden-state dimension is set to 500, with the latent variable z fed to decoder as input at every time step.

After the models converge, we randomly sample 5000 sentences from the test set and

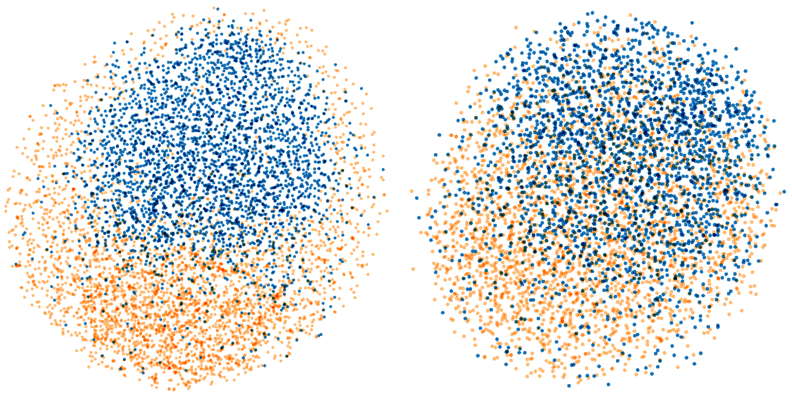

Figure 2.2: t-SNE embeddings of latent codes (left: DeConv-LVM, right: LSTM-LVM) for BookCorpus and arXiv sentences.

Model	# params	Time	KL	Acc
LSTM-LVM	$\sim$ 16 million	39m 41s	4.6	91.7
DeConv-LVM	$\sim$ 12 million	**8m 23s**	31.7	**96.2**

Table 2.2: Quantitative comparison between latent-variable models with LSTM and deconvolutional networks as the sentence decoder.

map their 500-dimensional latent embeddings, z, to a 2D vector using t-SNE [MH08]. The embedding plots for DeConv-LVM (left) and LSTM-LVM (right) are shown in Figure 2.2. For both cases, the plot shape of sampled latent embeddings is very close to a circle, which means the posterior distribution $p(z|x)$ matches the Gaussian prior $p(z)$ well. More importantly, when we use deconvolutional networks as the decoder, disentangled latent codes for the two writing styles can be clearly observed in the majority of prior space. This indicates that the semantic meanings of a sentence are encoded into the latent variable z, even when we train the model in an unsupervised manner. On the contrary, the latent codes of LSTM-LVM inferred for different writing styles tend to mix with each other, and cannot be separated as easily as in the case of Deconv-LVM, suggesting that less information may be encoded into the embeddings.

To better understand the advantages of deconvolutional networks as the decoder in the latent-variable models, we perform a quantitative comparison between the latent codes in DeConv-LVM and LSTM-LVM. In Table 2.2 we show the number of parameters, training time for 10,000 iterations, and the percentage of KL loss in the total loss for both models. Moreover, we extract sentence features from each model, and train a linear classifier on top, to distinguish between scientific and informal writing styles. The sentence embeddings are fixed during training, in order to elucidate the quality of latent codes learned in an unsupervised manner. 1000 sentences are sampled from the training set to learn the classifier and the classification accuracy is calculated on the whole test set. DeConv-LVM (96.2%) performs better than LSTM-LVM (91.7%), again indicating that the the latent codes of DeConv-LVM are more informative. This observation corresponds well with the fact that the percentage of KL loss in DeConv-LVM (31.7%) is much larger than in LSTM-LVM (4.6%), where larger KL divergence loss can be considered as a sign that more useful information has been encoded in the latent variable z [BVV$^+$16a, YHSBK17a]. Further, we observe that DeConv-LVM has relatively few parameters compared to LSTM-LVM, making it a promising latent-variable model for text.

2.4.3 Recognizing Textual Entailment (RTE)

Motivated by the superior performance of our deconvolutional latent-variable model on unsupervised learning, we further apply it to text sequence matching, in a semi-supervised scenario. We consider the task of recognizing text entailment on the Stanford Natural Language Inference (SNLI) dataset [BAPM15b].

To check the generalization ability of our latent variable learned, we experimented with different amounts of labeled training data (other sentence pairs in the training set are used as unlabeled data). The results are shown in Figure 2.3. Compared to the LSTM baseline models in [BAPM15b] and our basic CNN implementation, both our autoencoder and latent-variable models make use of the unlabeled data and achieve better results than

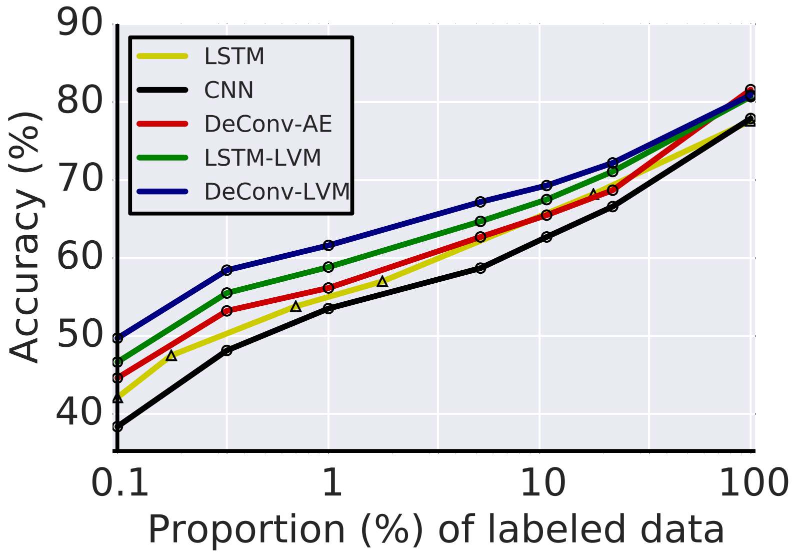

Figure 2.3: The performance of various models on SNLI dataset, with different amount of labeled data.

simply train an encoder network, *i.e.*, LSTM, CNN, only with the labeled data. More importantly, the DeConv-LVM we propose outperforms LSTM-LVM in all cases, consistent with previous observations that the latent variable z in our DeConv-LVM tends to be more informative. Note that when using all labeled data when training, DeConv-AE (81.6%) performs a bit better than DeConv-LVM (80.9%), which is not surprising since DeConv-LVM introduces a further constraint on the latent features learned (close to prior distribution) and may not be optimal when a lot of labeled data are available for training.

To directly compare with [KZR$^+$17] on semi-supervised learning experiments, we follow their experiment setup where 28k, 59k, 120k labeled examples are used for training. According to Table 2.3, it turns out that our DeConv-AE model is a competitive baseline, and outperform their LSTM-AE results. Moreover, our DeConv-LVM achieves even better results than DeConv-AE and LSTM-LVM, suggesting that the deconvolution-based latent-variable model we propose makes effective use of unsupervised information. Further, we see that the gap tends to be larger when the number of labeled data is smaller, further demonstrating that DeConv-LVM is a promising strategy to extract useful informa-

Model	28k	59k	120k
LSTM ([KZR$^+$17])	57.9	62.5	65.9
LSTM-AE ([KZR$^+$17])	59.9	64.6	68.5
LSTM-ADAE ([KZR$^+$17])	62.5	66.8	70.9
CNN (random)	58.7	62.7	65.6
CNN (Glove)	60.3	64.1	66.8
DeConv-AE	62.1	65.5	68.7
LSTM-LVM	64.7	67.5	71.1
DeConv-LVM	**67.2**	**69.3**	**72.2**

Table 2.3: Semi-supervised recognizing textual entailment accuracy on SNLI dataset, in percentage.

tion from unlabeled data.

2.4.4 Paraphrase Identification

We investigate our deconvolutional latent-variable model on the paraphrase identification task with the Quora Question Pairs dataset, following the same dataset split as [WHF17]. We consider cases where 1k, 5k, 10k, 25k labeled examples are used for training. As illustrated in Table 2.4, a CNN encoder with Glove pre-trained word embeddings consistently outperforms that with randomly initialized word embeddings, while the autoencoder model achieves better results than only training a CNN encoder, corresponding with findings in [DL15a].

More importantly, our latent-variable models show even higher accuracy than autoencoder models, demonstrating that they effectively utilize the information of unlabeled data and that they represent an effective strategy for paraphrase identification task. Our DeConv-LVM again performs better than LSTM-LVM in all cases, indicating that the deconvolutional decoder can leverage more benefits from the latent-variable model. However, we can also see the trend that with larger number of labeled data, the gaps between these

Model	1k	5k	10k	25k
CNN (random)	56.3	59.2	63.8	68.9
CNN (Glove)	58.5	62.4	66.1	70.2
LSTM-AE	59.3	63.8	67.2	70.9
DeConv-AE	60.2	65.1	67.7	71.6
LSTM-LVM	62.9	67.6	69.0	72.4
DeConv-LVM	**65.1**	**69.4**	**70.5**	**73.7**

Table 2.4: Paraphrase identification accuracy on Quora Question Pairs dataset, in percentages.

models are smaller. This may be attributed to the fact that when lots of labeled data are available, discriminative information tends be the dominant factor for better performance, while the information from unlabeled data becomes less important.

2.5 Related Work

The proposed framework is closely related to recent research on incorporating NVI into text modeling [BVV$^+$16a, MYB16a, XSDT17a, ZXS$^+$16a, SSL$^+$17c]. [BVV$^+$16a] presented the first attempt to utilize NVI for language modeling, but their results using an LSTM decoder were largely negative. [MYB16a] applied the NVI framework to an unsupervised bags-of-words model. However, from the perspective of text representation learning, their model ignores word-order information, which may be suboptimal for downstream supervised tasks. [XSDT17a] employed a variational autoencoder with the LSTM-LSTM architecture for semi-supervised sentence classification. However, as illustrated in our experiments, as well as in [YHSBK17a], the LSTM decoder is not the most effective choice for learning informative and discriminative sentence embeddings.

The NVI framework has also been employed for text-generation problems, such as machine translation [ZXS$^+$16a] and dialogue generation [SSL$^+$17c], with the motivation to improve the diversity and controllability of generated sentences. Our work is distinguished from this prior research in two principal respects: (*i*) We leveraged the NVI framework for latent variable models to text sequence matching tasks, due to its ability to take advantage of unlabeled data and learn robust sentence embeddings; (*ii*) we employed deconvolutional networks, instead of the LSTM, as the decoder (generative) network. We demonstrated the effectiveness of our framework in both unsupervised and supervised (including semi-supervised) learning cases.

2.6 Conclusion

We have presented a latent variable model for matching natural language sentences, with deconvolutional networks as the sequence encoder. We show that by jointly optimizing the variational lower bound and matching loss, the model is effective at inferring robust sentence representations for determining their semantic relationship, even with limited amount of labeled data. State-of-the-art experimental results on two semi-supervised sequence matching tasks are achieved, demonstrating the advantages of our approach. This work provides a promising strategy towards training effective and fast latent-variable models for text data.

Chapter 3

NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing

Semantic hashing has become a powerful paradigm for fast similarity search in many information retrieval systems. While fairly successful, previous techniques generally require two-stage training, and the binary constraints are handled *ad-hoc*. In this chapter, I will present an *end-to-end* Neural Architecture for Semantic Hashing (NASH), where the binary hashing codes are treated as *Bernoulli* latent variables. A neural variational inference framework is proposed for training, where gradients are directly backpropagated through the discrete latent variable to optimize the hash function. We also draw connections between proposed method and *rate-distortion theory*, which provides a theoretical foundation for the effectiveness of the proposed framework. Experimental results on three public datasets demonstrate that our method significantly outperforms several state-of-the-art models on both *unsupervised* and *supervised* scenarios.

3.1 Introduction

The problem of *similarity search*, also called *nearest-neighbor search*, consists of finding documents from a large collection of documents, or *corpus*, which are most similar to a query document of interest. Fast and accurate similarity search is at the core of many information retrieval applications, such as plagiarism analysis [SzEP07], collaborative filtering [Kor08], content-based multimedia retrieval [LSDJ06] and caching [PBC^{+}09]. Semantic hashing is an approach for fast similarity search [SH09, ZWCL10, WSSJ14]. By representing every document in the corpus as a similarity-preserving discrete (binary) *hashing code*, the similarity between two documents can be evaluated by simply calculating pairwise Hamming distances between hashing codes, *i.e.*, the number of bits that are different between two codes. Given that today, an ordinary PC is able to execute millions of Hamming distance computations in just a few milliseconds [ZWCL10], this semantic hashing strategy is very computationally attractive.

While considerable research has been devoted to text (semantic) hashing, existing approaches typically require two-stage training procedures. These methods can be generally divided into two categories: (i) binary codes for documents are first learned in an unsupervised manner, then l binary classifiers are trained via supervised learning to predict the l-bit hashing code [ZWCL10, XWT$^+$15]; (ii) continuous text representations are first inferred, which are binarized as a second (separate) step during testing [WZS13, CF17]. Because the model parameters are not learned in an end-to-end manner, these two-stage training strategies may result in suboptimal local optima. This happens because different modules within the model are optimized separately, preventing the sharing of information between them. Further, in existing methods, binary constraints are typically handled *ad-hoc* by truncation, *i.e.*, the hashing codes are obtained via direct binarization from continuous representations after training. As a result, the information contained in the continuous representations is lost during the (separate) binarization process. Moreover, training different modules (mapping and classifier/binarization) separately often requires additional hyperparameter tuning for each training stage, which can be laborious and time-consuming.

In this paper, we propose a simple and generic neural architecture for text hashing that learns binary latent codes for documents in an *end-to-end* manner. Inspired by recent advances in neural variational inference (NVI) for text processing [MYB16a, YHSBK17a, SZH$^+$18a], we approach semantic hashing from a generative model perspective, where binary (hashing) codes are represented as either *deterministic* or *stochastic* Bernoulli latent variables. The inference (encoder) and generative (decoder) networks are optimized jointly by maximizing a variational lower bound to the marginal distribution of input documents

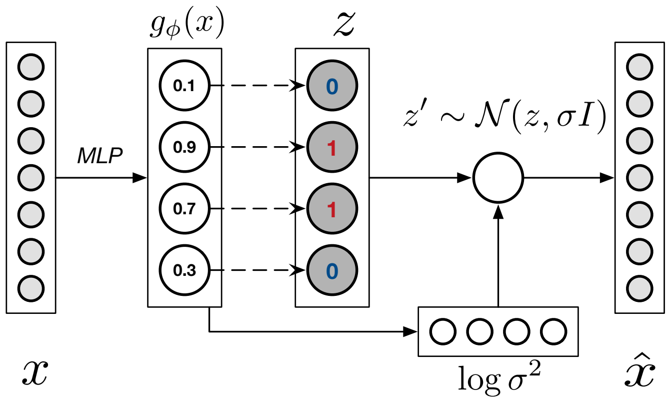

Figure 3.1: NASH for *end-to-end* semantic hashing. The inference network maps $x \rightarrow z$ using an MLP and the generative network recovers x as $z \rightarrow \hat{x}$.

(corpus). By leveraging a simple and method to estimate the gradients with respect to discrete (binary) variables, the loss term from the generative (decoder) network can be directly backpropagated into the inference (encoder) network to optimize the hash function.

Motivated by the *rate-distortion theory* [Ber03, TSCH17], we propose to inject data-dependent noise into the latent codes during the decoding stage, which adaptively accounts for the tradeoff between minimizing *rate* (number of bits used, or code length) and *distortion* (reconstruction error) during training. The connection between the proposed method and *rate-distortion theory* is further elucidated, providing a theoretical foundation for the ness of our framework.

Summarizing, the contributions of this paper are: (*i*) to the best of our knowledge, we present the first semantic hashing architecture that can be trained in an *end-to-end* manner; (*ii*) we propose a *neural variational inference* framework to learn compact (regularized) binary codes for documents, achieving promising results on both *unsupervised* and *supervised* text hashing; (*iii*) the connection between our method and *rate-distortion theory* is established, from which we demonstrate the advantage of injecting *data-dependent noise* into the latent variable during training.

3.2 The Proposed Method

3.2.1 Hashing under the NVI Framework

Inspired by the recent success of variational autoencoders for various NLP problems [MYB16a, BVV+15, YHSBK17a, MGB17b, SZH+18a, WGW+17], we approach the training of discrete (binary) latent variables from a generative perspective. Let x and z denote the input document and its corresponding binary hash code, respectively. Most of the previous text hashing methods focus on modeling the encoding distribution $p(z|x)$, or *hash function*, so the local/global pairwise similarity structure of documents in the original space is preserved in latent space [ZWCL10, WZS13, XWT+15, WSSJ14]. However, the generative (decoding) process of reconstructing x from binary latent code z, *i.e.*, modeling distribution $p(x|z)$, has been rarely considered. Intuitively, latent codes learned from a model that accounts for the generative term should naturally encapsulate key semantic information from x because the generation/reconstruction objective is a function of $p(x|z)$. In this regard, the generative term provides a natural training objective for semantic hashing.

We define a generative model that simultaneously accounts for both the encoding distribution, $p(z|x)$, and decoding distribution, $p(x|z)$, by defining approximations $q_\phi(z|x)$ and $q_\theta(x|z)$, via inference and generative networks, $g_\phi(x)$ and $g_\theta(z)$, parameterized by ϕ and θ, respectively. Specifically, $x \in \mathcal{Z}_+^{|V|}$ is the bag-of-words (count) representation for the input document, where $|V|$ is the vocabulary size. Notably, we can also employ other count weighting schemes as input features x, *e.g.*, the term frequency-inverse document frequency (TFIDF) [MPH08]. For the encoding distribution, a latent variable z is first inferred from the input text x, by constructing an inference network $g_\phi(x)$ to approximate the true posterior distribution $p(z|x)$ as $q_\phi(z|x)$. Subsequently, the decoder network $g_\theta(z)$ maps z back into input space to reconstruct the original sequence x as $\hat{x}$, approximating $p(x|z)$ as $q_\theta(x|z)$ (as shown in Figure 5.1). This *cyclic* strategy, $x \to z \to \hat{x} \approx x$, provides the latent variable z with a better ability to generalize [MYB16a].

To tailor the NVI framework for semantic hashing, we cast z as a binary latent variable and assume a multivariate Bernoulli prior on z: $p(z) \sim \text{Bernoulli}(\gamma) = \prod_{i=1}^{l} \gamma_i^{z_i}(1 - \gamma_i)^{1-z_i}$, where $\gamma_i \in [0, 1]$ is component i of vector γ. Thus, the encoding (approximate posterior) distribution $q_\phi(z|x)$ is restricted to take the form $q_\phi(z|x) = \text{Bernoulli}(h)$, where $h = \sigma(g_\phi(x))$, $\sigma(\cdot)$ is the sigmoid function, and $g_\phi(\cdot)$ is the (nonlinear) inference network specified as a multilayer perceptron (MLP). As illustrated in Figure 5.1, we can obtain samples from the Bernoulli posterior either *deterministically* or *stochastically*. Suppose z is a l-bit hash code, for the *deterministic* binarization, we have, for $i = 1, 2, ..., l$:

$$z_i = \mathbf{1}_{\sigma(g_\phi^i(x)) > 0.5} = \frac{\text{sign}(\sigma(g_\phi^i(x) - 0.5) + 1}{2}, \tag{3.1}$$

where z is the binarized variable, and z_i and $g_\phi^i(x)$ denote the i-th dimension of z and $g_\phi(x)$, respectively. The standard Bernoulli sampling in (3.1) can be understood as setting a hard threshold at 0.5 for each representation dimension, therefore, the binary latent code is generated deterministically. Another strategy to obtain the discrete variable is to binarize h in a *stochastic* manner:

$$z_i = \mathbf{1}_{\sigma(g_\phi^i(x)) > \mu_i} = \frac{\text{sign}(\sigma(g_\phi^i(x)) - \mu_i) + 1}{2}, \tag{3.2}$$

where $\mu_i \sim \text{Uniform}(0, 1)$. Because of this sampling process, we do not have to assume a pre-defined threshold value like in (3.1).

3.2.2 Training with Binary Latent Variables

To estimate the parameters of the encoder and decoder networks, we would ideally maximize the marginal distribution $p(x) = \int p(z)p(x|z)dz$. However, computing this marginal is intractable in most cases of interest. Instead, we maximize a variational lower bound.

This approach is typically employed in the VAE framework [KW13]:

$$\mathcal{L}_{\text{vae}} = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\theta(x|z)p(z)}{q_\phi(z|x)}\right], \tag{3.3}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)),$$

where the Kullback-Leibler (KL) divergence $D_{KL}(q_\phi(z|x)||p(z))$ encourages the approximate posterior distribution $q_\phi(z|x)$ to be close to the multivariate Bernoulli prior $p(z)$. In this case, $D_{KL}(q_\phi(z|x)|p(z))$ can be written in closed-form as a function of $g_\phi(x)$:

$$D_{KL} = g_\phi(x) \log \frac{g_\phi(x)}{\gamma}$$

$$+ (1 - g_\phi(x)) \log \frac{1 - g_\phi(x)}{1 - \gamma}. \tag{3.4}$$

Note that the gradient for the KL divergence term above can be evaluated easily.

For the first term in (4.1), we should in principle estimate the influence of μ_i in (3.2) on $q_\theta(x|z)$ by averaging over the entire (uniform) noise distribution. However, a closed-form distribution does not exist since it is not possible to enumerate all possible configurations of z, especially when the latent dimension is large. Moreover, discrete latent variables are inherently incompatible with backpropagation, since the derivative of the sign function is zero for almost all input values. As a result, the exact gradients of L_{vae} wrt the inputs before binarization would be essentially all zero.

To estimate the gradients for binary latent variables, we utilize the straight-through (ST) estimator, which was first introduced by [Hin12]. So motivated, the strategy here is to simply backpropagate through the hard threshold by approximating the gradient $\partial z/\partial \phi$

as 1. Thus, we have:

$$\begin{aligned}
&\frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{\partial \phi} \\
&\quad = \frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{dz} \frac{dz}{d\sigma(g_\phi^i(x))} \frac{d\sigma(g_\phi^i(x))}{d\phi} \\
&\quad \approx \frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{dz} \frac{d\sigma(g_\phi^i(x))}{d\phi} \qquad (3.5)
\end{aligned}$$

Although this is clearly a biased estimator, it has been shown to be a fast and efficient method relative to other gradient estimators for discrete variables, especially for the Bernoulli case [BLC13, HCS$^+$16, TSCH17]. With the ST gradient estimator, the first loss term in (4.1) can be backpropagated into the encoder network to fine-tune the hash function $g_\phi(x)$.

For the approximate generator $q_\theta(x|z)$ in (4.1), let x_i denote the one-hot representation of ith word within a document. Note that $x = \sum_i x_i$ is thus the bag-of-words representation for document x. To reconstruct the input x from z, we utilize a *softmax* decoding function written as:

$$q(x_i = w|z) = \frac{\exp(z^T E x_w + b_w)}{\sum_{j=1}^{|V|} \exp(z^T E x_j + b_j)}, \qquad (3.6)$$

where $q(x_i = w|z)$ is the probability that x_i is word $w \in V$, $q_\theta(x|z) = \prod_i q(x_i = w|z)$ and $\theta = \{E, b_1, \dots, b_{|V|}\}$. Note that $E \in \mathbb{R}^{d \times |V|}$ can be interpreted as a word embedding matrix to be learned, and $\{b_i\}_{i=1}^{|V|}$ denote bias terms. Intuitively, the objective in (5.4) encourages the discrete vector z to be close to the embeddings for every word that appear in the input document x. As shown in Section 3.4.3, meaningful semantic structures can be learned and manifested in the word embedding matrix E.

3.2.3 Injecting Data-dependent Noise to z

To reconstruct text data x from sampled binary representation z, a deterministic decoder is typically utilized [MYB16a, CF17]. Inspired by the success of employing stochastic decoders in image hashing applications [DGK+17, TSCH17], in our experiments, we found that injecting random Gaussian noise into z makes the decoder a more favorable regularizer for the binary codes, which in practice leads to stronger retrieval performance. Below, we invoke the *rate-distortion theory* to perform some further analysis, which leads to interesting findings.

Learning binary latent codes z to represent a continuous distribution $p(x)$ is a classical information theory concept known as *lossy source coding*. From this perspective, semantic hashing, which compresses an input document into compact binary codes, can be casted as a conventional *rate-distortion tradeoff* problem [TSCH17, BLS16]:

$$\min \underbrace{-\log_2 R(z)}_{\text{Rate}} + \beta \cdot \underbrace{D(x, \hat{x})}_{\text{Distortion}}, \tag{3.7}$$

where *rate* and *distortion* denote the code length, *i.e.*, the number of bits used, and the distortion introduced by the encoding/decoding sequence, respectively. Further, $\hat{x}$ is the reconstructed input and β is a hyperparameter that controls the tradeoff between the two terms.

Considering the case where we have a Bernoulli *prior* on z as $p(z) \sim \text{Bernoulli}(\gamma)$, and x conditionally drawn from a Gaussian distribution $p(x|z) \sim \mathcal{N}(Ez, \sigma^2 I)$. Here, $E = \{e_i\}_{i=1}^{|V|}$, where $e_i \in \mathbb{R}^d$ can be interpreted as a *codebook* with $|V|$ *codewords*. In our case, E corresponds to the *word embedding matrix* as in (5.4).

For the case of stochastic latent variable z, the objective function in (4.1) can be written in a form similar to the *rate-distortion* tradeoff:

$$\min \mathbb{E}_{q_\phi(z|x)} \left[\underbrace{-\log q_\phi(z|x)}_{\text{Rate}} + \underbrace{\frac{1}{2\sigma^2}}_{\beta} \underbrace{||x - Ez||_2^2}_{\text{Distortion}} + C \right], \tag{3.8}$$

where C is a constant that encapsulates the prior distribution $p(z)$ and the Gaussian distribution normalization term. Notably, the trade-off hyperparameter $\beta = \sigma^{-2}/2$ is closely related to the variance of the distribution $p(x|z)$. In other words, by controlling the variance σ, the model can adaptively explore different trade-offs between the *rate* and *distortion* objectives. However, the optimal trade-offs for distinct samples may be different.

Inspired by the observations above, we propose to inject data-dependent noise into latent variable z, rather than to setting the variance term σ^2 to a fixed value [DGK$^+$17, TSCH17]. Specifically, $\log \sigma^2$ is obtained via a one-layer MLP transformation from $g_\phi(x)$. Afterwards, we sample z' from $\mathcal{N}(z, \sigma^2 I)$, which then replace z in (5.4) to infer the probability of generating individual words (as shown in Figure 5.1). As a result, the variances are different for every input document x, and thus the model is provided with additional flexibility to explore various trade-offs between *rate* and *distortion* for different training observations. Although our decoder is not a strictly Gaussian distribution, as in (5.4), we found empirically that injecting data-dependent noise into z yields strong retrieval results, see Section 3.4.1.

3.2.4 Supervised Hashing

The proposed Neural Architecture for Semantic Hashing (NASH) can be extended to supervised hashing, where a mapping from latent variable z to labels y is learned, here parametrized by a two-layer MLP followed by a fully-connected softmax layer. To allow the model to explore and balance between maximizing the variational lower bound in (4.1) and minimizing the discriminative loss, the following joint training objective is employed:

$$\mathcal{L} = -\mathcal{L}_{\text{vae}}(\theta, \phi; x) + \alpha \mathcal{L}_{\text{dis}}(\eta; z, y). \tag{3.9}$$

where η refers to parameters of the MLP classifier and α controls the relative weight between the variational lower bound ($\mathcal{L}_{\text{vae}}$) and discriminative loss ($\mathcal{L}_{\text{dis}}$), defined as the cross-entropy loss. The parameters $\{\theta, \phi, \eta\}$ are learned end-to-end *via* Monte Carlo estimation.

3.3 Experimental Setup

3.3.1 Datasets

We use the following three standard publicly available datasets for training and evaluation: (i) *Reuters21578*, containing 10,788 news documents, which have been classified into 90 different categories. (ii) *20Newsgroups*, a collection of 18,828 newsgroup documents, which are categorized into 20 different topics. (iii) TMC (stands for SIAM text mining competition), containing air traffic reports provided by NASA. TMC consists 21,519 training documents divided into 22 different categories. To make direct comparison with prior works, we employed the TFIDF features on these datasets supplied by [CF17], where the vocabulary sizes for the three datasets are set to 10,000, 7,164 and 20,000, respectively.

3.3.2 Training Details

For the inference networks, we employ a feed-forward neural network with 2 hidden layers (both with 500 units) using the ReLU non-linearity activation function, which transform the input documents, *i.e.*, TFIDF features in our experiments, into a continuous representation. Empirically, we found that stochastic binarization as in (3.2) shows stronger performance than deterministic binarization, and thus use the former in our experiments. However, we further conduct a systematic ablation study in Section 3.4.2 to compare the two binarization strategies.

Our model is trained using Adam [KB14a], with a learning rate of 1×10^{-3} for all parameters. We decay the learning rate by a factor of 0.96 for every 10,000 iterations. Dropout [SHK$^+$14a] is employed on the output of encoder networks, with the rate selected from $\{0.7, 0.8, 0.9\}$ on the validation set. To facilitate comparisons with previous methods, we set the dimension of z, *i.e.*, the number of bits within the hashing code) as 8, 16, 32, 64, or 128.

3.3.3 Baselines

We evaluate the ness of our framework on both unsupervised and supervised semantic hashing tasks. We consider the following *unsupervised* baselines for comparisons: Locality Sensitive Hashing (LSH) [DIIM04], Stack Restricted Boltzmann Machines (S-RBM) [SH09], Spectral Hashing (SpH) [WTF09], Self-taught Hashing (STH) [ZWCL10] and Variational Deep Semantic Hashing (VDSH) [CF17].

For supervised semantic hashing, we also compare NASH against a number of baselines: Supervised Hashing with Kernels (KSH) [LWJ$^+$12], Semantic Hashing using Tags and Topic Modeling (SHTTM) [WZS13] and Supervised VDSH [CF17]. It is worth noting that unlike all these baselines, our NASH model is trained end-to-end in one-step.

3.3.4 Evaluation Metrics

To evaluate the hashing codes for similarity search, we consider each document in the testing set as a query document. Similar documents to the query in the corresponding training set need to be retrieved based on the Hamming distance of their hashing codes, *i.e.* number of different bits. To facilitate comparison with prior work [WZS13, CF17], the performance is measured with precision. Specifically, during testing, for a query document, we first retrieve the 100 nearest/closest documents according to the Hamming distances of the corresponding hash codes (i.e., the number of different bits). We then examine the percentage of documents among these 100 retrieved ones that belong to the same label (topic) with the query document (we consider documents having the same label as relevant pairs). The ratio of the number of relevant documents to the number of retrieved documents (fixed value of 100) is calculated as the precision score. The precision scores are further averaged over all test (query) documents.

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
NASH-N	0.7352	0.7904	0.8297	0.8086	0.7867
NASH-DN	**0.7470**	**0.8013**	**0.8418**	**0.8297**	**0.7924**

Table 3.1: Precision of the top 100 retrieved documents on *Reuters* dataset (*Unsupervised hashing*).

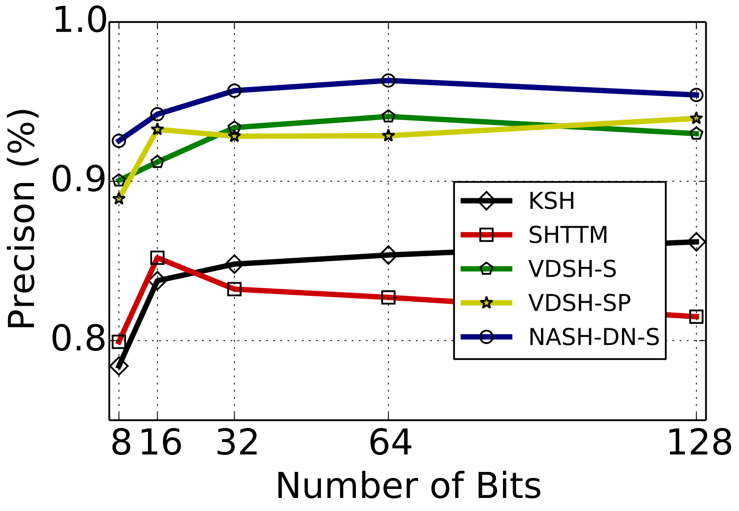

Figure 3.2: Precision of the top 100 retrieved documents on *Reuters* dataset (*Supervised hashing*), compared with other supervised baselines.

3.4 Experimental Results

We experimented with four variants for our NASH model: (*i*) NASH: with deterministic decoder; (*ii*) NASH-N: with *fixed* random noise injected to decoder; (*iii*) NASH-DN: with *data-dependent* noise injected to decoder; (*iv*) NASH-DN-S: NASH-DN with supervised information during training.

3.4.1 Semantic Hashing Evaluation

Table 3.1 presents the results of all models on Reuters dataset. Regarding unsupervised semantic hashing, all the NASH variants consistently outperform the baseline methods by a substantial margin, indicating that our model makes the most use of unlabeled data and manage to assign similar hashing codes, *i.e.*, with small Hamming distance to each other, to documents that belong to the same label. It can be also observed that the injection of noise into the decoder networks has improved the robustness of learned binary representations, resulting in better retrieval performance. More importantly, by making the variances of noise adaptive to the specific input, our NASH-DN achieves even better results, compared with NASH-N, highlighting the importance of exploring/learning the trade-off between rate and distortion objectives by the data itself. We observe the same trend and superiority of our NASH-DN models on the other two benchmarks, as shown in Tables 3.3 and 3.4.

Word	**weapons**	**medical**	**companies**	**define**	**israel**	**book**
NASH	gun	treatment	company	definition	israeli	books
	guns	disease	market	defined	arabs	english
	weapon	drugs	afford	explained	arab	references
	armed	health	products	discussion	jewish	learning
	assault	medicine	money	knowledge	jews	reference
NVDM	guns	medicine	expensive	defined	israeli	books
	weapon	health	industry	definition	arab	reference
	gun	treatment	company	printf	arabs	guide
	militia	disease	market	int	lebanon	writing
	armed	patients	buy	sufficient	lebanese	pages

Table 3.2: The five nearest words in the semantic space learned by NASH, compared with the results from NVDM [MYB16a].

Another observation is that the retrieval results tend to drop a bit when we set the length of hashing codes to be 64 or larger, which also happens for some baseline models. This phenomenon has been reported previously in [WKC12, LWJ+12, WZS13, CF17],

Method	8 bits	16 bits	32 bits	64 bits	128 bits
Unsupervised Hashing					
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
NASH-N	0.3903	0.5213	0.5987	0.5143	0.4776
NASH-DN	**0.4040**	**0.5310**	**0.6225**	0.5377	0.4945
Supervised Hashing					
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**

Table 3.3: Precision of the top 100 retrieved documents on *20Newsgroups* dataset.

and the reasons could be twofold: (i) 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 [WKC12]; (ii) considering the size of training data, it is likely that the model may overfit with long hash codes [CF17]. However, even with longer hashing codes, our NASH models perform stronger than the baselines in most cases (except for the 20Newsgroups dataset), suggesting that NASH can ly allocate documents to informative/meaningful hashing codes even with limited training data.

We also evaluate the ness of NASH in a *supervised* scenario on the Reuters dataset, where the label or topic information is utilized during training. As shown in Figure 3.2, our NASH-DN-S model consistently outperforms several supervised semantic hashing baselines, with various choices of hashing bits. Notably, our model exhibits higher Top-100 retrieval precision than VDSH-S and VDSH-SP, proposed by [CF17]. This may be at-

Method	8 bits	16 bits	32 bits	64 bits	128 bits
Unsupervised Hashing					
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
NASH-N	0.6233	0.6759	0.7201	0.6877	0.6314
NASH-DN	**0.6358**	**0.6956**	**0.7327**	**0.7010**	**0.6325**
Supervised Hashing					
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**

Table 3.4: Precision of the top 100 retrieved documents on *TMC* dataset.

tributed to the fact that in VDSH models, the continuous embeddings are not optimized with their future binarization in mind, and thus could hurt the relevance of learned binary codes. On the contrary, our model is optimized in an end-to-end manner, where the gradients are directly backpropagated to the inference network (through the binary/discrete latent variable), and thus gives rise to a more robust hash function.

3.4.2 Ablation study

The effect of stochastic sampling

As described in Section 5.3.4, the binary latent variables z in NASH can be either deterministically (via (3.1)) or stochastically (via (3.2)) sampled. We compare these two types of binarization functions in the case of unsupervised hashing. As illustrated in Figure 3.3, stochastic sampling shows stronger retrieval results on all three datasets, indicating that endowing the sampling process of latent variables with more stochasticity improves the learned representations.

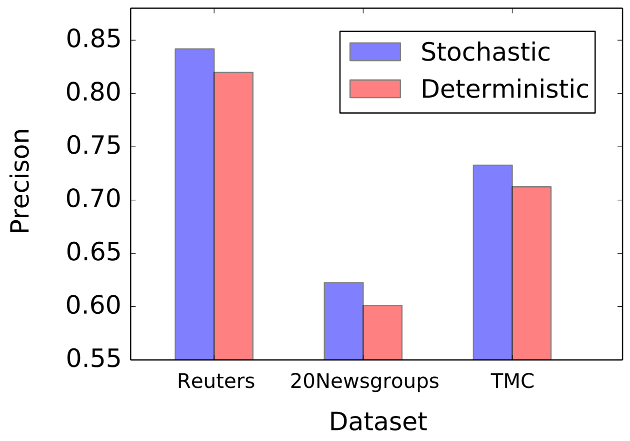

Figure 3.3: The precisions of the top 100 retrieved documents for NASH-DN with stochastic or deterministic binary latent variables.

The effect of encoder/decoder networks

Category	Title/Subject	8-bit code	16-bit code
Baseball	*Dave Kingman for the hall of fame*	11101001	0010110100000110
	Time of game	11111001	0010100100000111
	Game score report	11101001	0010110100000110
	Why is Barry Bonds not batting 4th?	11101101	0011110100000110
Electronics	*Building a UV flashlight*	10110100	0010001000101011
	How to drive an array of LEDs	10110101	0010001000101001
	2% silver solder	11010101	0010001000101011
	Subliminal message flashing on TV	10110100	0010011000101001

Table 3.5: Examples of learned compact hashing codes on *20Newsgroups* dataset.

Under the variational framework introduced here, the encoder network, *i.e.*, hash function, and decoder network are jointly optimized to abstract semantic features from documents. An interesting question concerns what types of network should be leveraged for each part of our NASH model. In this regard, we further investigate the effect of using an encoder or decoder with different non-linearity, ranging from a linear transformation to two-layer MLPs. We employ a base model with an encoder of two-layer MLPs and a linear decoder (the setup described in Section 5.3.4), and the ablation study results are shown in Table 3.6.

Network	**Encoder**	**Decoder**
linear	0.5844	0.6225
one-layer MLP	0.6187	0.3559
two-layer MLP	0.6225	0.1047

Table 3.6: Ablation study with different encoder/decoder networks.

It is observed that for the encoder networks, increasing the non-linearity by stacking MLP layers leads to better empirical results. In other words, endowing the hash function with more modeling capacity is advantageous to retrieval tasks. However, when we employ a non-linear network for the decoder, the retrieval precision drops dramatically. It is worth noting that the only difference between linear transformation and one-layer MLP is whether a non-linear activation function is employed or not.

This observation may be attributed the fact that the decoder networks can be considered as a similarity measure between latent variable z and the word embeddings E_k for every word, and the probabilities for words that present in the document is maximized to ensure that z is informative. As a result, if we allow the decoder to be too expressive (*e.g.*, a one-layer MLP), it is likely that we will end up with a very flexible similarity measure but relatively less meaningful binary representations. This finding is consistent with several image hashing methods, such as SGH [DGK$^+$17] or binary autoencoder [CPR15], where a linear decoder is typically adopted to obtain promising retrieval results. However, our experiments may not speak for other choices of encoder-decoder architectures, *e.g.*, LSTM-based sequence-to-sequence models [SVL14] or DCNN-based autoencoder [ZSW$^+$17].

3.4.3 Qualitative Analysis

Analysis of Semantic Information

To understand what information has been learned in our NASH model, we examine the matrix $E \in \mathbb{R}^{d \times l}$ in (5.4). Similar to [MYB16a, LL12], we select the 5 nearest words according to the word vectors learned from NASH and compare with the corresponding results from NVDM.

As shown in Table 3.2, although our NASH model contains a binary latent variable, rather than a continuous one as in NVDM, it also ly group semantically-similar words together in the learned vector space. This further demonstrates that the proposed generative framework manages to bypass the binary/discrete constraint and is able to abstract useful semantic information from documents.

Case Study

In Table 3.5, we show some examples of the learned binary hashing codes on *20Newsgroups* dataset. We observe that for both 8-bit and 16-bit cases, NASH typically compresses documents with shared topics into very similar binary codes. On the contrary, the hashing codes for documents with different topics exhibit much larger Hamming distance. As a result, relevant documents can be efficiently retrieved by simply computing their Hamming distances.

3.5 Conclusion

This paper presents a first step towards *end-to-end* semantic hashing, where the binary/discrete constraints are carefully handled with an gradient estimator. A neural variational framework is introduced to train our model. Motivated by the connections between the proposed method and *rate-distortion theory*, we inject data-dependent noise into the Bernoulli latent variable at the training stage. The effectiveness of our framework is demonstrated with extensive experiments.

Chapter 4

Towards Generating Long and Coherent Text with Multi-Level Latent Variable Models

Variational autoencoders (VAEs) have received much attention recently as an end-to-end architecture for text generation with latent variables. However, previous works typically focus on synthesizing relatively short sentences (up to 20 words), and the posterior collapse issue has been widely identified in text-VAEs. In this chapter, I will propose to leverage several multi-level structures to learn a VAE model for generating *long*, and *coherent* text. In particular, a hierarchy of stochastic layers between the encoder and decoder networks is employed to abstract more informative and semantic-rich latent codes. Besides, we utilize a multi-level decoder structure to capture the coherent long-term structure inherent in long-form texts, by generating intermediate sentence representations as high-level *plan vectors*. Extensive experimental results demonstrate that the proposed multi-level VAE model produces more coherent and less repetitive long text compared to baselines as well as can mitigate the posterior-collapse issue.

4.1 Introduction

The variational autoencoder (VAE) for text [BVV+16b] is a generative model in which a stochastic latent variable provides additional information to modulate the sequential text-generation process. VAEs have been used for various text processing tasks [SSB17b, ZZE17, KWM+18, DLH+18, HGOL18, SSC+18, XD18, WGX+19b]. While most recent work has focused on generating relatively short sequences (*e.g.,* a single sentence or multiple sentences up to around twenty words), generating long-form text (*e.g.,* a single or multiple paragraphs) with deep latent-variable models has been less explored.

***flat*-VAE** (standard)	***multilevel*-VAE** (our model)
i went here for a grooming and a dog . it was very good . **the owner is very nice and friendly . the owner is really nice and friendly** . i don t know what they are doing .	i have been going to this nail salon for over a year now . the last time i went there . the stylist was nice . but the lady who did my nails . she was very rude and did not have the best nail color i once had .
the staff is very friendly and helpful . **the only reason i** can t give them 5 stars . **the only reason i** am giving the ticket is because of the ticket . **can t help but** the staff is so friendly and helpful . **can t help but** the parking lot is just the same .	i am a huge fan of this place . my husband and i were looking for a place to get some good music . this place was a little bit pricey . but i was very happy with the service . the staff was friendly .

Table 4.1: Comparison of samples generated from two generative models on the Yelp reviews dataset.

Recurrent Neural Networks (RNNs) [BCB15, CAR16] have mainly been used for most text VAE models [BVV$^+$16b]. However, it may be difficult to scale RNNs for *long-form* text generation, as they tend to generate text that is repetitive, ungrammatical, self-contradictory, overly generic and often lacking coherent long-term structure [HBF$^+$18]. Two samples of text generated using standard VAE with an RNN decoder is shown in Table 4.1.

In this work, we propose various multi-level network structures for the VAE model (*ml*-VAE), to address coherency and repetitiveness challenges associated with long-form text generation. To generate globally-coherent long text sequences, it is desirable that both the *higher-level* abstract features (*e.g.*, topic, sentiment, etc.) and *lower-level* fine-granularity details (*e.g.*, specific word choices) of long text can be leveraged by the generative network. It's difficult for a standard RNN to capture such structure and learn to *plan-ahead*. To improve the model's plan-ahead capability for capturing long-term dependency, following [RER$^+$18a], our first multi-level structure defines a hierarchical RNN decoder as the generative network that learns *sentence-* and *word-level* representations. Rather than using the latent code to initialize the RNN decoder directly, we found it more effective when first passing the latent code to a higher-level (sentence) RNN decoder, that outputs an embedding for the lower-level (word) RNN decoder that generates words. Since the low-level decoder network cannot fall back on autoregression, it gains a stronger reliance on the latent code to reconstruct the sequences.

Prior work has found that VAE training often suffers from *posterior collapse*, in which the model ignores the latent code [BVV$^+$16b]. This issue is related to the fact that the decoder network is usually parametrized with an autoregressive neural network, such as RNNs with teacher forcing scheme [BVV$^+$16b, YHSBK17b, GSC$^+$17, SSB17b, SZH$^+$18b]. Several strategies have been proposed (see optimization challenges in Section 4.4) to make the decoder less autoregressive, so less *contextual information* is utilized by the decoder network [YHSBK17b, SZH$^+$18b]. We argue that learning more informative latent codes can enhance the generative model without the need to lessen the contextual information. We propose leveraging a hierarchy of latent variables between the convolutional inference (encoder) networks and a multi-level recurrent generative network (decoder). With multiple stochastic layers, the prior of bottom-level latent variable is inferred from the data, rather than fixed as a standard Gaussian distribution as in typical VAEs [KW13]. The induced *latent code* distribution at the bottom level can be perceived as a Gaussian mixture, and thus is endowed with more flexibility to abstract meaningful features from the input text. While recent work has explored structures for more informative latent codes [KWM$^+$18, GCHK18], *ml*-VAE is conceptually simple and easy to implement.

We evaluate *ml*-VAE on language modeling, unconditional and conditional text generation tasks. We show substantial improvements against several baseline methods in terms of *perplexity* on language modeling and quality of generated samples based on BLEU statistics and human evaluation.

4.2 Variational Autoencoder (VAE)

Let $\boldsymbol{x}$ denote a text sequence, which consists of L tokens, *i.e.*, $x_1, x_2, ..., x_L$. A VAE encodes the text $\boldsymbol{x}$ using a recognition (encoder) model, $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, parameterizing an approx-

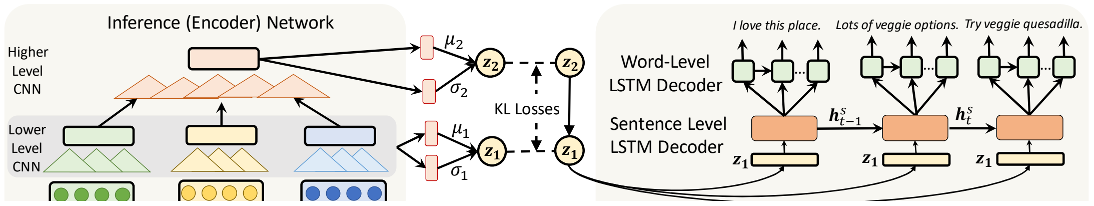

Figure 4.1: The proposed *multi-level* VAE with *double* latent variables (*ml*-VAE-D).

imate posterior distribution over a continuous latent variable $\boldsymbol{z}$ (whose prior is typically chosen as standard diagonal-covariance Gaussian). $\boldsymbol{z}$ is sampled stochastically from the posterior distribution, and text sequences $\boldsymbol{x}$ are generated conditioned on $\boldsymbol{z}$, via a generative (decoder) network, denoted as $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. A variational lower bound is typically used to estimate the parameters [KW13]:

$$\begin{aligned}\mathcal{L}_{\text{vae}} &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right], \qquad (4.1)\\ &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})),\end{aligned}$$

This lower bound is composed of a reconstruction loss (first term) that encourages the inference network to encode information necessary to generate the data and a KL regularizer (second term) to push $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ towards the prior $p(\boldsymbol{z})$.

Although VAEs have been shown to be effective in a wide variety of text processing tasks (see related work), there are two challenges associated with generating longer sequences with VAEs: (i) they lack a long-term planning mechanism, which is critical for generating *semantically-coherent* long texts [SKS+17]; and (ii) *posterior collapse* issue. Concerning (ii), it was demonstrated in [BVV+16b] that due to the autoregressive nature of the RNN, the decoder tends to ignore the information from $\boldsymbol{z}$ entirely, resulting in an extremely small KL term (see Section 4.4).

4.3 Multi-Level Generative Networks

4.3.1 Single Latent Variable (*ml*-VAE-S:)

Our first multi-level model improves upon standard VAE models by introducing a *plan-ahead* ability to sequence generation. Instead of directly making word-level predictions only conditioned on the semantic information from $\boldsymbol{z}$, a series of *plan vectors* are first generated based upon $\boldsymbol{z}$ with a *sentence-level* LSTM decoder [LLJ15b]. Our hypothesis is that an explicit design of (inherently hierarchical) paragraph structure can capture sentence-level coherence and potentially mitigate repetitiveness. Intuitively, when predicting each token, the decoder can use information from both the words generated previously and from sentence-level representations.

An input paragraph consist of M sentences, and each sentence t has N_t words, t=1,$\ldots$, M. To generate the plan vectors, the model first samples a latent code $\boldsymbol{z}$ through a one-layer multi-layered perceptron (MLP), with ReLU activation functions, to obtain the starting state of the sentence-level LSTM decoder. Subsequent sentence representations, namely the *plan vectors*, are generated with the sentence-level LSTM in a sequential manner:

$$\boldsymbol{h}_t^s = \text{LSTM}^{\text{sent}}(\boldsymbol{h}_{t-1}^s, \boldsymbol{z}), \tag{4.2}$$

The latent code $\boldsymbol{z}$ can be considered as a paragraph-level abstraction, relating to information about the semantics of each generated subsequence. Therefore we input $\boldsymbol{z}$ at each time step of the sentence-level LSTM, to predict the sentence representation. Our single-latent-variable model sketched in Figure 4.3 of supplementary material.

The generated sentence-level plan vectors are then passed onto the word-level LSTM decoder to generate the words for each sentence. To generate each word of a sentence t, the corresponding *plan vector*, $\boldsymbol{h}_t^s$, is concatenated with the word embedding of the previous word and fed to LSTM^{word} at every time step [1]. Let $w_{t,i}$ denote the i-th token of the t-th

[1]We use teacher-forcing during training and greedy decoding at test time.

sentence This process can be expressed as (for $t = 1, 2, ..., M$ and $i = 1, 2, 3, ..., N_t$):

$$\boldsymbol{h}^w_{t,i} = \text{LSTM}^{\text{word}}(\boldsymbol{h}^w_{t,i-1}, \boldsymbol{h}^s_t, \boldsymbol{W_e}[w_{t,i-1}]), \tag{4.3}$$

$$p(w_{t,i}|w_{t,<i}, \boldsymbol{h}^s_t) = \text{softmax}(\boldsymbol{V}\boldsymbol{h}^w_{t,i}), \tag{4.4}$$

The initial state $\boldsymbol{h}^w_{t,0}$ of LSTM^{word} is inferred from the corresponding plan vector via an MLP layer. $\boldsymbol{V}$ represents the weight matrix for computing distribution over words, and $\boldsymbol{W_e}$ are word embeddings to be learned. For each sentence, once the special _END token is generated, the word-level LSTM stops decoding [2]. LSTM^{word} decoder parameters are shared for each generated sentence.

4.3.2 Double Latent Variables (*ml*-VAE-D):

Similar architectures of our single latent variable *ml*-VAE-S model have been applied recently for multi-turn dialog response generation [SSL$^+$17a, PCK18], mainly focusing on short (one-sentence) response generation. Different from these works, our goal is to generate *long text* which introduces additional challenges to the hierarchical generative network. We hypothesize that with the two-level LSTM decoder embedded into the VAE framework, the load of capturing global and local semantics are handled differently than the *flat*-VAEs [CKS$^+$16]. While the multi-level LSTM decoder can capture relatively detailed information (*e.g.*, word-level (local) coherence) via the word- and sentence-level LSTM networks, the latent codes of the VAE are encouraged to abstract more global and high-level semantic features of multiple sentences of long text.

Our double latent variable extension, *ml*-VAE-D, is shown in Figure 5.1. The inference network encodes upward through each latent variable to infer their posterior distributions, while the generative network samples downward to obtain the distributions over the latent variables. The distribution of the latent variable at the bottom is inferred from the top-layer latent codes, rather than fixed (as in a standard VAE model). This also introduces flexibility

[2]Each sentence is padded with an _END token.

to the model to abstract useful high-level features [GKA$^+$16], which can then be leveraged by the multi-level LSTM network. Without loss of generality, here we choose to employ a two-layer hierarchy of latent variables, where the bottom and top layers are denoted as $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$, respectively, which can be easily extended to multiple latent-variable layers.

Another important advantage of multi-layer latent variables in the VAE framework is related to the *posterior collapse* issue. Even though the single latent variable model (*ml*-VAE-S) defines a multi-level LSTM decoder, the posterior collapse can still exist since the LSTM decoder can still ignore the latent codes due to its autoregressive property. With the hierarchical latent variables, we propose a novel strategy to mitigate this problem, by making less restrictive assumptions regarding the prior distribution of the latent variable. Our model yields a larger KL loss term relative to *flat*-VAEs, indicating more informative latent codes.

The posterior distributions over the latent variables are assumed to be conditionally independent given the input x. We can represent the joint posterior distribution of the two latent variables as [3]:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z_1}, \boldsymbol{z_2}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z_2}|\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z_1}|\boldsymbol{x}) \tag{4.5}$$

Concerning the generative network, the latent variable at the bottom is sampled conditioned on the one at the top. Thus, we have:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z_1}, \boldsymbol{z_2}) = p_{\boldsymbol{\theta}}(\boldsymbol{z_2})p_{\boldsymbol{\theta}}(\boldsymbol{z_1}|\boldsymbol{z_2}) \tag{4.6}$$

$D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$, the second term of the VAE objective, then becomes the KL divergence between joint posterior and prior distributions of the two latent variables. Under the assumptions of (4.5) and (4.6), the variational lower bound yields:

$$\mathcal{L}_{\text{vae}} = \mathbb{E}_{q(\boldsymbol{z_1}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z_1})] - D_{\text{KL}}(q(\boldsymbol{z_1}, \boldsymbol{z_2}|\boldsymbol{x})||p(\boldsymbol{z_1}, \boldsymbol{z_2})) \tag{4.7}$$

[3]We assume $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ to be independent on the encoder side, since this specification will yield a closed-form expression for the KL loss between $p_{\boldsymbol{\theta}}(\boldsymbol{z_1}, \boldsymbol{z_2})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z_1}, \boldsymbol{z_2}|\boldsymbol{x})$.

Abbreviarting $p_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{\phi}}$ with p and q, we get:

$$\begin{aligned}
&D_{\mathrm{KL}}(q(\boldsymbol{z_1}, \boldsymbol{z_2}|\boldsymbol{x})||p(\boldsymbol{z_1}, \boldsymbol{z_2})) \\
&= \int q(\boldsymbol{z_2}|\boldsymbol{x})q(\boldsymbol{z_1}|\boldsymbol{x}) \log \frac{q(\boldsymbol{z_2}|\boldsymbol{x})q(\boldsymbol{z_1}|\boldsymbol{x})}{p(\boldsymbol{z_2})p(\boldsymbol{z_1}|\boldsymbol{z_2})} d\boldsymbol{z_1} d\boldsymbol{z_2} \\
&= \int_{\boldsymbol{z_1}, \boldsymbol{z_2}} [q_{\boldsymbol{\phi}}(\boldsymbol{z_2}|\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z_1}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z_1}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z_1}|\boldsymbol{z_2})} \\
&\quad + q(\boldsymbol{z_2}|\boldsymbol{x})q(\boldsymbol{z_1}|\boldsymbol{x}) \log \frac{q(\boldsymbol{z_2}|\boldsymbol{x})}{p(\boldsymbol{z_2})}] d\boldsymbol{z_1} d\boldsymbol{z_2} \\
&= \mathbb{E}_{q(\boldsymbol{z_2}|\boldsymbol{x})}[D_{KL}(q(\boldsymbol{z_1}|\boldsymbol{x})||p(\boldsymbol{z_1}|\boldsymbol{z_2}))] \\
&\quad + D_{KL}(q(\boldsymbol{z_2}|\boldsymbol{x})||p(\boldsymbol{z_2}))
\end{aligned} \tag{4.8}$$

The left-hand side of (4.8) is the abbreviation of $D_{\mathrm{KL}}(q_\phi(\boldsymbol{z_1}, \boldsymbol{z_2}|\boldsymbol{x})||p(\boldsymbol{z_1}, \boldsymbol{z_2}))$. Given the Gaussian assumption for both the prior and posterior distributions, both KL divergence terms can be written in closed-form.

To abstract meaningful representations from the input paragraphs, we choose a hierarchical CNN architecture for the inference/encoder networks for both single and double latent variable models. We use sentence-level CNNs to encode each sentence into a fixed-length vector, which are then aggregated and send to paragraph-level CNN encoder. The inference networks parameterizing $q(\boldsymbol{z_1}|\boldsymbol{x})$ and $q(\boldsymbol{z_2}|\boldsymbol{x})$ share the same parameters as the lower-level CNN.

The single-variable *ml*-VAE-S model feeds the paragraph feature vector into the linear layers to infer the mean and variance of the latent variable z. In the double-variable model *ml*-VAE-D, the feature vector is transformed with two MLP layers, and then is used to compute the mean and variance of the top-level latent variable.

4.4 Related Work

Optimization Challenges. The “posterior collapse” issue associated with training text-VAEs was first outlined by [BVV$^+$16b]. They used two strategies, *KL divergence annealing* and *word dropout*, however, none of them help to improve the perplexity compared to a plain neural language model. [YHSBK17b] argue that the small KL term relates to the strong autoregressive nature of an LSTM generative network, and they proposed to utilize a dilated CNN as a decoder to improve the informativeness of the latent variable. [ZLE18] proposed to augment the VAE training objective with an additional mutual information term. This yields an intractable integral in the case where the latent variables are continuous. Recent work [HSNB19, FLL$^+$19] has shown that advanced scheduling can mitigate the posterior collapse issue. We instead introduce more flexible priors and hierarchical encoder and decoder structures to deal with posterior collapse.

Hierarchical Structures. Natural language is inherently hierarchical (characters form a word, words form a sentence, sentences form a paragraph, paragraphs from a document, *etc*.). Previous work used multi-level LSTM encoders [YYD$^+$16] or hierarchical autoencoders [LLJ15a] to learn hierarchical representations for long text or defined a stochastic latent variable for each sentence at decoding time [SSL$^+$17a]. In contrast, our model encodes the entire paragraph into one *single* latent variable. The latent variable learned in our model relates more to the global semantic information of a paragraph, whereas those in [SSL$^+$17a] mainly contain the local information of a specific sentence. **(author?)**[PCK18] introduced a variational hierarchical conversational model (VHCR) with global and local latent variables. They generate local/utterance variables condintioned on the global latent variable, assuming standard dialog-covariance Gaussian for both latent variables. In contrast, both our latent variables in *ml*-VAE-D are designed to contain global information. *ml*-VAE learns the prior of the bottom-level latent variable from the data, yielding more flexible prior relative to a fixed prior and promising results in mitigating the issue of “posterior collapse” in the experiments. The responses in VHCR are generated conditionally on the latent variables and context, while our *ml*-VAE-D model captures the underlying data distribution of the entire paragraph in the bottom latent variable (z_1), so the global latent variable contains more information.

4.5 Experiments

4.5.1 Experimental Setup

Datasets We conducted experiments on both generic (unconditional) long-form text generation and conditional paragraph generation (with additional text input as auxiliary information). For the former, we use two datasets: Yelp Reviews [ZZL15] and arXiv Abstracts. For the conditional-generation experiments, we consider the task of synthesizing a paper abstract conditioned on the paper title (with the arXiv Abstracts dataset)[4]. Details on dataset statistics and model architectures are provided in the supplementary material.

Baselines We implement the following langauge modeling baselines: language model with a flat LSTM decoder (*flat*-LM), VAE with a flat LSTM decoder (*flat*-VAE), and language model with a multi-level LSTM decoder (*ml*-LM)[5].

For generic text generation, we build models using two recently proposed generative models as baselines: Adversarial Autoencoders (AAE) [MSJG15] and Adversarially-Regularized Autoencoders (ARAE) [ZKZ$^+$18]. Instead of penalizing the KL divergence term, AAE introduces a discriminator network to match the prior and posterior distributions of the latent variable. AARE model extends AAE by introducing Wassertein GAN loss [ACB17] and a stronger generator network. We build two variants of our multi-level

[4]Our goal is to analyze if the proposed architecture can discover different concepts with the hierarchical decoding and latent code structures, thus we use the arxiv dataset with indicated domains for demonstration purposes. We leave the common summarization datasets for future research.

[5]We only experimented with state of the art models with similar architectures to our models, since our goal is to investigate the impact of hiararhical VAE structure on the text generation. More efficient new encoder and decoder architectures such as non-autoregressive models is a direction for extending this work.

Model	Yelp			arXiv		
	NLL	KL	PPL	NLL	KL	PPL
flat-LM	162.6	-	48.0	218.7	-	57.6
flat-VAE	$\leq$ 163.1	0.01	$\leq$ 49.2	$\leq$ 219.5	0.01	$\leq$ 58.4
ml-LM	162.4	-	47.9	219.3	-	58.1
ml-VAE-S	$\leq$ 160.8	3.6	$\leq$ 46.6	$\leq$ 216.8	5.3	$\leq$ 55.6
ml-VAE-D	$\leq$ **160.2**	6.8	$\leq$ **45.8**	$\leq$ **215.6**	12.7	$\leq$ **54.3**

Table 4.2: Language modeling results on Yelp and arXiv data. Upper block are baselines, and lower are our models.

VAE models: single latent variable *ml*-VAE-S and double latent variable *ml*-VAE-D. Our code will be released to encourage future research.

4.5.2 Language Modeling Results

We report negative log likelihood (NLL) and perplexity (PPL) results on Yelp and arXiv datasets. Following [BVV+16b, YHSBK17b, KWM+18], we use the KL loss term to measure the extent of *"posterior collapse"*.

As shown in Table 4.2, the standard *flat*-VAE on Yelp dataset yields a KL divergence term very close to zero, indicating that the generative model makes negligible use of the information from latent variable z. The *flat*-VAE model obtains slightly worse NNL and PPL relative to a flat LSTM-based language model. With multi-level LSTM decoder, our *ml*-VAE-S yields increased KL divergence, demonstrating that the VAE model tends to leverage more information from the latent variable in the decoding stage. The PPL of *ml*-VAE-S is also decreased from 47.9 to 46.6 (compared to *ml*-LM), indicating that the sampled latent codes improve word-level predictions.

Our double latent variable model, *ml*-VAE-D, exhibits an even larger KL divergence cost term (increased from 3.6 to 6.8) than single latent variable model, indicating that more information from the latent variable has been utilized by the generative network. This may be due to the fact that the latent variable priors of the *ml*-VAE-D model are inferred from the data, rather than a fixed standard Gaussian distribution. As a result, the model is endowed with more flexibility to encode informative semantic features in the latent variables, yet matching their posterior distributions to the corresponding priors. *ml*-VAE-D achieves the best PPL results on both datasets (on the arXiv dataset, our hierarchical decoder outperforms the *ml*-LM by reducing the PPL from 58.1 down to 54.3).

4.5.3 Unconditional Text Generation

We evaluate the quality of generated paragraphs as follows. We randomly sample 1000 latent codes and send them to all trained generative models to generate text. We use corpus-level BLEU score [PRWZ02] to quantitatively evaluate the generated paragraphs. Following strategy in [YZWY17, ZGF$^+$17a] we use the entire test set as the reference for each generated text, and get the average BLEU scores[6] over 1000 generated sentences for each model.

The results are in Table 4.3. VAE tends to be a stronger baseline for paragraph generation, exhibiting higher corpus-level BLEU scores than both AAE and ARAE. This observation is consistent with the results in [CSAF18] in Table 4.3. The VAE with multi-level decoder demonstrates better BLEU scores than the one with a flat decoder, indicating that the plan-ahead mechanism associated with the hierarchical decoding process indeed benefits the sampling quality. *ml*-VAE-D exhibits slightly better results than *ml*-VAE-S. We attribute this to the more flexible prior distribution of *ml*-VAE-D, which improves the ability of inference networks to extract semantic features from a paragraph, yielding more informative latent codes.

We visualize the learnt latent variables to analyze if our models can extract global

[6]Being interested in longer text generation, we evaluate our models on the n-gram reconscturion ability (where n>1).

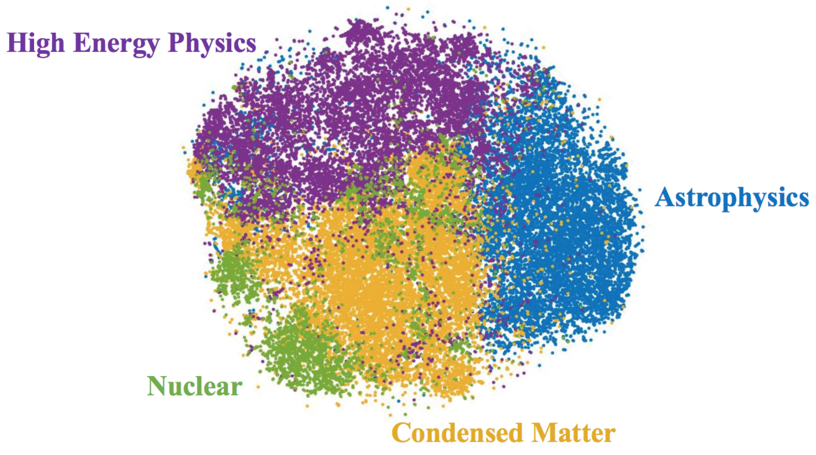

Figure 4.2: t-SNE visualization of the learned latent codes.

features. Using the arXiv dataset, we select the most frequent four article topics and re-train our *ml*-VAE-D model on the corresponding abstracts in an unsupervised way (no topic information is used). We sample bottom-level latent codes from the learned model and plot them with t-SNE in Figure 4.2. Each point indicates one paper abstract and the color of each point indicates the topic it belongs to. The embeddings of the same label are very close in the 2-D plot, while those with different labels are relatively farther away from each other. The embeddings of the *High Energy Physics* and *Nuclear* topic abstracts are meshed, which is expected since these two topics are semantically highly related. The inference network can extract meaningful global patterns from the input paragraph.

In Table 4.1 two samples of generations from *flat*-VAE and *ml*-VAE-D are shown. Compared to our hierarchical model, a flat decoder with a flat VAE exibits repetitions as well as suffers from uninformative sentences. The hierarchical model generates reviews that contain more information with less repetitions (word or semantic semantic repetitions), and tend to be semantically-coherent.

Diversity of Generated Paragraphs We evaluate the diversity of random samples from a trained model, since one model might generate realistic-looking sentences while suffer-

Model	**Yelp**			**arXiv**		
	B-2	**B-3**	**B-4**	**B-2**	**B-3**	**B-4**
*A*RAE	0.684	0.524	0.350	0.624	0.475	0.305
*A*AE	0.735	0.623	0.383	0.729	0.564	0.342
flat-VAE	0.855	0.705	0.515	0.784	0.625	0.421
ml-VAE-S	0.901	0.744	0.531	0.821	**0.663**	0.447
ml-VAE-D	**0.912**	**0.755**	**0.549**	**0.825**	0.657	**0.460**

Table 4.3: Evaluation results for generated sequences by our models and baselines on corpus-level BLEU scores (B-n denotes the corpus-level BLEU-n score.)

ing from severe mode collapse (*i.e.*, low diversity). We use three metrics to measure the diversity of generated paragraphs: Self-BLEU scores [ZLZ$^+$18], unique n-grams [FGD18] and the entropy score [ZGG$^+$18]. For a set of sampled sentences, the Self-BLEU metric is the BLEU score of each sample with respect to all other samples as the reference (the numbers over all samples are then averaged); the unique score computes the percentage of *unique* n-grams within all the generated reviews; and the entropy score measures how evenly the empirical n-gram distribution is for a given sentence, which does not depend on the size of testing data, as opposed to unique scores. In all three metrics, lower is better. We randomly sample 1000 reviews from each model.

The results are shown in Table 4.5. A small self-BLEU score together with a large BLEU score can justify the effectiveness of a model, *i.e.*, being able to generate realistic-looking as well as diverse samples. Among all the VAE variants, *ml*-VAE-D shows the smallest BLEU score and largest unique n-grams percentage, demonstrating the effectiveness of hieararhically structured generative networks as well as latent variables. Even though AAE and ARAE yield better diversity according to both metrics, their corpus-level BLEU scores are much worse relative to *ml*-VAE-D. We leverage human evaluation for

we study the effect of disorder on the dynamics of a two-dimensional electron gas in a two-dimensional optical lattice , we show that the superfluid phase is a phase transition , we also show that , in the presence of a magnetic field , the vortex density is strongly enhanced .
in this work we study the dynamics of a colloidal suspension of frictionless , the capillary forces are driven by the UNK UNK , when the substrate is a thin film , the system is driven by a periodic potential, we also study the dynamics of the interface between the two different types of particles .

Table 4.4: Generated arXiv abstracts from *ml*-VAE-D model.

Model	**Yelp**						
	B-2	**B-3**	**B-4**	**2gr**	**3gr**	**4gr**	**Etp-2**
ARAE	0.725	0.544	0.402	36.2	59.7	75.8	7.551
AAE	0.831	0.672	0.483	33.2	57.5	71.4	6.767
flat-VAE	0.872	0.755	0.617	23.7	48.2	69.0	6.793
ml-VAE-S	0.865	0.734	0.591	28.7	50.4	70.7	6.843
ml-VAE-D	0.851	0.723	0.579	30.5	53.2	72.6	6.926

Table 4.5: Evaluation of diversity of 1000 generated sentences on self-BLEU scores (B-n), unique n-gram percentages (ngr), 2-gram entropy score.

further comparison.

Human Evaluation We conducted human evaluations using Amazon Mechanical Turk to assess the coherence and non-redundancy properties of our proposed models. Given a pair of generated reviews, the judges are asked to select their preferences (no difference between the two reviews is also an option) according to the following four evaluation criteria: *fluency & grammar*, *consistency*, *non-redundancy*, and *overall*. We compare generated text from our *ml*-VAE-D againts flat-VAE, AAE and real samples from the test set. Details are provided in the supplementary material.

As shown in Table 4.7, *ml*-VAE generates superior human-looking samples compared to *flat*-VAE on the Yelp Reviews dataset. Even though both models underperform when compared against the ground-truth real reviews, *ml*-VAE was rated higher in comparison to *flat*-VAE (raters find *ml*-VAE closer to human-generated than the *flat*-VAE) in all the criteria evaluation criteria. When compared against AAE baseline models using the same data preprocessing steps and hyperparameters, *ml*-VAE again produces more grammatically-correct and semantically-coherent samples. The human evaluations correlate with the automatic metrics, which indicate that our *ml*-VAE is actually generating more coherent stories than the baseline models. We leave further evaluations using embedding based metrics as a possible extension to our work.

4.5.4 Conditional Paragraph Generation

We consider the task of generating an abstract of a paper based on the corresponding title. The same arXiv dataset is utilized, where when training the title and abstract are given as paired text sequences. The title is used as input of the inference network. For the generative network, instead of reconstructing the same input (*i.e.*, title), the paper abstract is employed as the target for decoding. We compare the *ml*-VAE-D model against *ml*-LM. We observe that the *ml*-VAE-D model achieves a test perplexity of 55.7 (with a KL term of 2.57), smaller that the test perplexity of *ml*-LM (58.1), indicating that the information from the title is used by the generative network to facilitate the decoding process. Generated abstract samples from *ml*-VAE-D model are shown in Table 4.6.

4.5.5 Analysis

The Continuity of Latent Space Following [BVV$^+$16b], we measure the continuity of the learned latent space. We randomly sample two points from the prior latent space (denoted as A and B) and generate sentences based on the equidistant intermediate points along the linear trajectory between A and B. As shown in Table 4.8, these intermediate samples are all realistic-looking reviews that are syntactically and semantically reasonable,

Title: *Magnetic quantum phase transitions of the antiferromagnetic - Heisenberg model*
We study the phase diagram of the model in the presence of a magnetic field, The model is based on the action of the Polyakov loop, We show that the model is consistent with the results of the first order perturbation theory.
Title: *Kalman Filtering With UNK Over Wireless UNK Channels*
The Kalman filter is a powerful tool for the analysis of quantum information, which is a key component of quantum information processing, However, the efficiency of the proposed scheme is not well understood .

Table 4.6: Conditionally generated arXiv paper abstracts from *ml*-VAE-D model based on a given title.

Model	Grammar.	Cons.	Non-Red.	Overall
ml-VAE	52.0	55.0	53.7	60.0
flat-VAE	30.0	33.0	27.7	32.3
ml-VAE	75.3	86.0	76.7	86.0
AAE	13.3	10.3	15.0	12.0
flat-VAE	19.7	18.7	14.3	19.0
Real data	61.7	74.7	74.3	77.7
ml-VAE	28.0	26.3	25.0	30.3
Real data	48.6	58.7	49.0	61.3

Table 4.7: Human evaluations. Each block is a head-to-head comparison of two models on grammatically, consistency, and non-redundancy.

A	the service was great, the receptionist was very friendly and the place was clean, we waited for a while, and then our room was ready .
•	same with all the other reviews, this place is a good place to eat, i came here with a group of friends for a birthday dinner, we were hungry and decided to try it, we were seated promptly.
•	this place is a little bit of a drive from the strip, my husband and i were looking for a place to eat, all the food was good, the only thing i didn t like was the sweet potato fries.
•	this is not a good place to go, the guy at the front desk was rude and unprofessional, it s a very small room, and the place was not clean.
•	service was poor, the food is terrible, when i asked for a refill on my drink, no one even acknowledged me, they are so rude and unprofessional.
B	how is this place still in business, the staff is rude, no one knows what they are doing, they lost my business .

Table 4.8: Intermediate sentences are produced from linear transition between two points in the latent space and sending them to the generator network.

demonstrating the smoothness of the learned VAE latent space. Interestingly, we even observe that the generated sentences gradually transit from positive to negative sentiment along the linear trajectory. To validate that the sentences are not generated by simply retrieving the training data, we find the closest instance, among the entire training set, for each generated review. Details of the results can be found in the supplementary material (Table 4.12).

Attribute Vector Arithmetic We conduct an experiment to alter the sentiments of reviews with an *attribute vector*. We encode the reviews of the Yelp Review training dataset with positive sentiment and sample a latent code for each review and measure the mean latent vector. The mean latent vector of the negative reviews are computed in the same way. We subtract the negative mean vector from the positive mean vector to obtain the "sentiment attribute vector". Next, for evaluation, we randomly sample 1000 reviews with negative sentiment and add the "sentiment attribute vector" to their latent codes. The manipulated latent vectors are then fed to the hierarchical decoder to produce the transferred

Original: you have no idea how badly i want to like this place, they are incredibly vegetarian vegan friendly , i just haven t been impressed by anything i ve ordered there , even the chips and salsa aren t terribly good , i do like the bar they have great sangria but that s about it .
Transferred: this is definitely one of my favorite places to eat in vegas , they are very friendly and the food is always fresh, i highly recommend the pork belly , everything else is also very delicious, i do like the fact that they have a great selection of salads .

Table 4.9: An example sentiment transfer result with attribute vector arithmetic. More examples can be found in the supplementary material (Table 4.13).

sentences, hypothesizing that they will convey positive sentiment.

As shown in Table 4.9, the original sentences have been successfully manipulated to positive sentiment with the simple attribute vector operation. However, the specific contents of the reviews are not fully retained. One interesting future direction is to decouple the style and content of long-form texts to allow *content-preserving* attribute manipulation. We employed a CNN sentiment classifier to evaluate the sentiment of manipulated sentences. The classifier is trained on the entire training set and achieves a test accuracy of 94.2%. With this pre-trained classifier, 83.4% of the transferred reviews are predicted as positive-sentiment, indicating that "attribute vector arithmetic" consistently produces the intended manipulation of sentiment.

4.6 Conclusion

We introduce a hierarchically-structured variational autoencoder for long text generation. It consists of a multi-level LSTM generative network to model the semantic coherence at both the word- and sentence-levels. A hierarchy of stochastic layers is employed, where the priors of the latent variables are learned from the data. Consequently, more informative latent codes are manifested, and the generated samples also exhibit superior quality relative to those from several baseline methods.

4.7 Supplementary

4.7.1 Datasets & Model Details

In the following, we provide details of data pre-processing and the experimental setups used in the experiments. For both Yelp Reviews and arXiv Abstracts datasets, we truncate the original paragraph to the first five sentences (split by punctuation marks including *comma*, *period* and *point* symbols), where each sentence contains at most 25 words. Therefore, each paragraph has at most 125 words. We remove those sentences that contain less than 30 words. The statistics of both datasets are detailed in Table 4.10. The average length of paragraphs considered here are much larger than previous generative models for text [BVV$^+$16b, YZWY17, HYL$^+$17, ZGF$^+$17a], since these works considered text sequences that contain only one sentence with at most twenty words.

Dataset	Train	Test	Vocabulary	Aver. Length
Yelp Reviews	244748	18401	12461	48
arXiv Abstracts	504268	28016	32487	59

Table 4.10: Summary statistics for the datasets used in the generic text generation experiments.

In all the VAE and extentions, the dimension of the latent variable z is set to 300. The dimensions of both the sentence-level and word-level LSTM decoders are set to 512. For the generative networks, to infer the bottom-level latent variable (*i.e.*, modeling $p(z_1|z_2)$), we first feed the sampled latent codes from z_2 to two MLP layers, which is followed by two linear transformation to infer the mean and variance of z_1, respectively.

The model is trained using Adam [KB14b] with a learning rate of 3×10^{-4} for all parameters, with a decay rate of 0.99 for every 3000 iterations. Dropout [SHK$^+$14b] is employed on both word embedding and latent variable layers, with rates selected from $\{0.3, 0.5, 0.8\}$ on the validation set. We set the mini-batch size to 128. Following [BVV$^+$16b] we adopt the KL cost annealing strategy to stabilize training: the KL cost term is increased linearly to 1 until 10,000 iterations. All experiments are implemented in Tensorflow [ABC$^+$16], using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

4.7.2 Additional Generated Samples from *ml*-VAE-D vs *flat*-VAE

We provide additional examples for the comparison between *ml*-VAE-D vs *flat*-VAE in Table 4.11, as a continuation of Table 4.1.

4.7.3 Retrieved closest training instances of generated samples (Yelp Reviews Dataset)

We provide samples of retrieved instances from the Yelp Review training dataset which are closest to the generated samples. Table 4.12 shows the closest training samples of each generated Yelp review. The first column indicates the intermediate generated sentences produced from linear transition from a point A to another point B in the prior latent space. The second column on the right are the real sentences retrieved from the training set that are closest to the ones generated on the left (determined by BLEU-2 score). We can see that the retrieved training data is quite different from the generated samples, indicating that our model is indeed generating samples that it has never seen during training.

4.7.4 Human evaluation setup and details

Some properties of the generated paragraphs, such as (topic) coherence or non-redundancy, can not be easily measured by automated metrics. Therefore, we conduct human evaluation based on 100 samples randomly generated by each model (the models are trained on the Yelp Reviews dataset for this evaluation). We consider *flat*-VAE, adversarial autoencoders (AAE) and real samples from the test set to compare with our proposed *ml*-VAE-D model. The same hyperparameters are employed for the different model variants to ensure fair

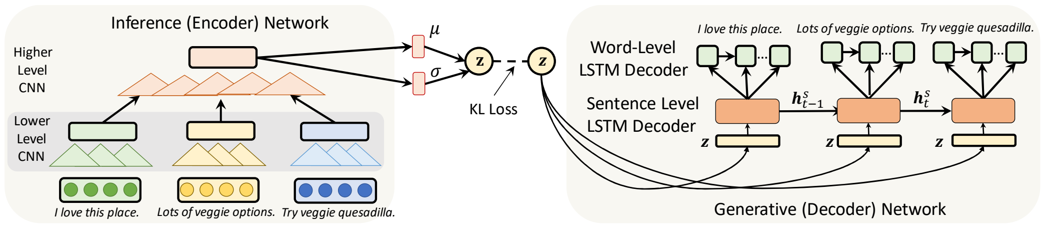

Figure 4.3: Schematic diagram of the proposed multi-level VAE with single latent variable.

comparison. We evaluate the quality of these generated samples with a blind heads-to-head comparison using Amazon Mechanical Turk. Given a pair of generated reviews, the judges are asked to select their preferences ("no difference between the two reviews" is also an option) according to the following 4 evaluation criteria: (1) *fluency & grammar*, the one that is more grammatically correct and fluent; (2) *consistency*, the one that depicts a sequence of topics and events that is more consistent; (3) *non-redundancy*, the one that is better at non-redundancy (if a review repeats itself, this can be taken into account); and (4) *overall*, the one that more effectively communicates reasonable content. These different criteria help to quantify the impact of the hierarchical structures employed in our model, while the non-redundancy and consistency metrics could be especially correlated with the model's plan-ahead abilities. The generated paragraphs are presented to the judges in a random order and they are not told the source of the samples. Each sample is rated by three judges and the results are averaged across all samples and judges.

4.7.5 More Samples on Attribute Vector Arithmetic

We provide more samples for sentiment manipulation, where we intend to alter sentiment of negative Yelp reviews with "attribute vector arithmetic", as a continuation of Table 4.9.

4.7.6 Comparison with the "utterance drop" strategy

To resolve the "posterior collapse" issue of training textual VAEs, [PCK18] also introduced a strategy called *utterance drop* (u.d). Specifically, they proposed to weaken the autoregressive power of hierarchical RNNs by dropping the utterance encoder vector with a certain probability. To investigate the effectiveness of their method relative to our strategy of employing a hierarchy of latent variables, we conduct a comparative study. Particularly, we utilize *ml*-VAE-S as the baseline model and apply the two strategies to it respectively. The corresponding results on language modeling (Yelp dataset) are shown in Table 4.14. Their u.d strategy indeed allows better usage of the latent variable (indicated by a larger KL divergence value). However, the NLL of the language model becomes even worse, possibly due to the weakening of the decoder during training (similar observations have also been reported in Table 2 of [PCK18]). Our hierarchical prior strategy yields larger KL terms as well as lower NNL value, indicating the advantage of our strategy to mitigate the "posterior collapse" issue.

Model	**NLL**	**KL**	**PPL**
ml-VAE-S	160.8	3.6	46.6
ml-VAE-S (with u.d)	161.3	5.6	47.1
ml-VAE-D	160.2	6.8	45.8

Table 4.14: Comparison with the *utterance drop* strategy.

***ml*-VAE**	***flat*-VAE**
i would give this place zero stars if i could , the guy who was working the front desk was rude and unprofessional , i have to say that i was in the wrong place , and i m not sure what i was thinking , this is not a good place to go to .	this is a great little restaurant in vegas , i had the shrimp scampi and my wife **had the shrimp scampi**, and my husband **had the shrimp scampi** , it was delicious , i **had the shrimp scampi** which was delicious and seasoned perfectly .
my wife and i went to this place for dinner , we were seated immediately , the food was good , i ordered the shrimp and grits , which was the best part of the meal .	**very good chinese food, very good chinese food**, the service was very slow, i guess that s what they were doing, very slow to get a quick meal.
we got a gift certificate from a store, we walked in and were greeted by a young lady who was very helpful and friendly, so we decided to get a cut, I was told that they would be ready in 15 minutes.	we go there for eakfast, i ve been here 3 times and it s always good, the hot dogs are delicious, and **the hot dogs are delicious**, i ve been there for eakfast and it is so good.
the place was packed, chicken was dry, tasted like a frozen hot chocolate, others were just so so, i wouldn t recommend this place.	do not go here, their food is terrible, they were very slow, in my opinion.
went today with my wife, and received a coupon for a free appetizer, we were not impressed, we both ordered the same thing, and **we were not impressed**.	the wynn is a great place to eat, the food was great and i had the linguine, and it was so good, i **had the linguine** and clams, (i was so excited to try it).
recently visited this place for the first time, i live in the area and have been looking for a good local place to eat, we stopped in for a quick bite and a few beers, always a nice place to sit and relax, wonderful and friendly staffs.	i came here for a quick bite before heading to a friend s recommendation, the place was packed, but the food was delicious, i am a fan of the place, and the place is **packed** with a lot of people.
best haircut i ve had in years, friendly staff and great service, he made sure that i was happy with my hair cut, just a little pricey but worth it, she is so nice and friendly.	had a great experience here today, the delivery was friendly and efficient and the food was good, i would recommend this place to anyone who will work in the future, will be back again.
great place to go for a date night, first time i went here, service is good, the staff is friendly, 5 stars for the food.	best place to get in vegas, ps the massage here is awesome, if you want to spend your money, then go there, ps **the massage is great**.

Table 4.11: Samples randomly generated from *ml*-VAE-D and *flat*-VAE, which are both trained on the Yelp review dataset.

	Generated samples	**Closest instance (in the training dataset)**
A	the service was great, the receptionist was very friendly and the place was clean, we waited for a while, and then our room was ready .	i ve only been here once myself , and i wasn t impressed , the service was great , staff was very friendly and helpful , we waited for nothing
●	same with all the other reviews, this place is a good place to eat, i came here with a group of friends for a birthday dinner, we were hungry and decided to try it, we were seated promptly.	i really love this place , red robin alone is a good place to eat , but the service here is great too not always easy to find , we were seated promptly , ought drinks promptly and our orders were on point .
●	this place is a little bit of a drive from the strip, my husband and i were looking for a place to eat, all the food was good, the only thing i didn t like was the sweet potato fries.	after a night of drinking , we were looking for a place to eat , the only place still open was the grad lux , its just like a cheesecake factory , the food was actually pretty good .
●	this is not a good place to go, the guy at the front desk was rude and unprofessional, it s a very small room, and the place was not clean.	the food is very good , the margaritas hit the spot , and the service is great , the atmosphere is a little cheesy but overall it s a great place to go .
●	service was poor, the food is terrible, when i asked for a refill on my drink, no one even acknowledged me, they are so rude and unprofessional.	disliked this place , the hostess was so rude , when i asked for a booth , i got attitude , a major .
B	how is this place still in business, the staff is rude, no one knows what they are doing, they lost my business .	i can t express how awful this store is , don t go to this location , drive to any other location , the staff is useless , no one knows what they are doing .

Table 4.12: Using the *ml*-VAE-D model trained on the Yelp Review dataset, intermediate sentences are produced from linear transition between two points in the prior latent space.

Original: papa j s is expensive and inconsistent , the ambiance is nice but it doesn t justify the prices , there are better restaurants in carnegie .	**Transferred**: love the food , the prices are reasonable and the food is great , it s a great place to go for a quick bite .
Original: i had a lunch there once , the food is ok but it s on the pricy side , i don t think i will be back .	**Transferred**: i had a great time here , the food is great and the prices are reasonable , i ll be back .
Original: i have to say that i write this review with much regret , because i have always loved papa j s , but my recent experience there has changed my mind a bit , from the minute we were seated , we were greeted by a server that was clearly inexperienced and didn t know the menu .	**Transferred**: i have to say , the restaurant is a great place to go for a date , my girlfriend and i have been there a few times , on my last visit , we were greeted by a very friendly hostess .
Original: a friend recommended this to me , and i can t figure out why , the food was underwhelming and pricey , the service was fine , and the place looked nice .	**Transferred**: a friend of mine recommended this place , and i was so glad that i did try it , the service was great , and the food was delicious .
Original: this is a small , franchise owned location that caters to the low income in the area , selection is quite limited throughout the store with limited quantities on the shelf of the items they do carry , because of the area in which it is located , the store is not 24 hours as most giant eagle s seem to be .	**Transferred**: this is a great little shop, easy to navigate , and they are always open , their produce is always fresh , the store is clean and the staff is friendly .

Table 4.13: Sentiment transfer results with attribute vector arithmetic.

Original: you have no idea how badly i want to like this place, they are incredibly vegetarian vegan friendly , i just haven t been impressed by anything i ve ordered there , even the chips and salsa aren t terribly good , i do like the bar they have great sangria but that s about it .
Transferred: this is definitely one of my favorite places to eat in vegas , they are very friendly and the food is always fresh, i highly recommend the pork belly , everything else is also very delicious, i do like the fact that they have a great selection of salads .
Original: my boyfriend and i are in our 20s , and have visited this place multiple times , after our visit yesterday , i don t think we ll be back , when we arrived we were greeted by a long line of people waiting to buy game cards .
Transferred: my boyfriend and i have been here twice , and have been to the one in gilbert several times too , since my first visit , i don t think i ve ever had a bad meal here , the servers were very friendly and helpful .

Table 4.15: An example sentiment transfer result with attribute vector arithmetic.

Chapter 5

Learning Compressed Sentence Representations for On-Device Text Processing

5.1 Introduction

Learning general-purpose sentence representations from large training corpora has received widespread attention in recent years. The learned sentence embeddings can encapsulate rich prior knowledge of natural language, which has been demonstrated to facilitate a variety of downstream tasks (*without* fine-tuning the encoder weights). The generic sentence embeddings can be trained either in an unsupervised manner [KZS$^+$15a, HCK16, JBS17, GPH$^+$17, LL18, PGJ18], or with supervised tasks such as paraphrase identification [WBGL16], natural language inference [CKS$^+$17], discourse relation classification [NBG17], machine translation [WG18], *etc*.

Significant effort has been devoted to designing better training objectives for learning sentence embeddings. However, prior methods typically assume that the general-purpose sentence representations are *continuous* and *real-valued*. However, this assumption is sub-optimal from the following perspectives: *i*) the sentence embeddings require large storage or memory footprint; *ii*) it is computationally expensive to retrieve semantically-similar sentences, since every sentence representation in the database needs to be compared, and the inner product operation is computationally involved. These two disadvantages hinder the applicability of generic sentence representations to mobile devices, where a relatively tiny memory footprint and low computational capacity are typically available [RK18].

In this paper, we aim to mitigate the above issues by binarizing the continuous sentence embeddings. Consequently, the embeddings require much smaller footprint, and similar sentences can be obtained by simply selecting those with closest binary codes in the Hamming space [KC18]. One simple idea is to naively binarize the continuous vectors by setting a hard threshold. However, we find that this strategy leads to significant performance drop in the empirical results. Besides, the dimension of the binary sentence embeddings cannot be flexibly chosen with this strategy, further limiting the practice use of the direct binarization method.

In this regard, we propose three alternative strategies to parametrize the transformation from pre-trained generic continuous embeddings to their binary forms. Our exploration spans from *simple* operations, such as a random projection, to *deep* neural network models, such as a *regularized* autoencoder. Particularly, we introduce a semantic-preserving objective, which is augmented with the standard autoenoder architecture to encourage abstracting informative binary codes. InferSent [CKS$^+$17] is employed as the testbed sentence embeddings in our experiments, but the binarization schemes proposed here can easily be extended to other pre-trained general-purpose sentence embeddings. We evaluate the quality of the learned general-purpose binary representations using the SentEval toolkit [CKS$^+$17]. It is observed that the inferred binary codes successfully maintain the semantic features contained in the continuous embeddings, and only lead to around 2% performance drop on a set of downstream NLP tasks, while requiring merely 1.5% memory footprint of their continuous counterparts.

Moreover, on several sentence matching benchmarks, we demonstrate that the relatedness between a sentence pair can be evaluated by simply calculating the Hamming distance between their binary codes, which perform on par with or even superior than measuring the cosine similarity between continuous embeddings (see Table 5.1). Note that computing the Hamming distance is much more computationally efficient than the inner product operation in a continuous space. We further perform a K-nearest neighbor sentence retrieval experiment on the SNLI dataset [BAPM15c], and show that those semantically-similar sentences can indeed be efficiently retrieved with off-the-shelf binary sentence representations. Summarizing, our contributions in this paper are as follows:

i) to the best of our knowledge, we conduct the first systematic exploration on learning general-purpose *binarized* (memory-efficient) sentence representations, and four different strategies are proposed;

ii) an autoencoder architecture with a carefully-designed *semantic-preserving loss* exhibits strong empirical results on a set of downstream NLP tasks;

iii) more importantly, we demonstrate, on several sentence-matching datasets, that simply evaluating the *Hamming distance* over binary representations performs on par or even better than calculating the *cosine similarity* between their continuous counterparts (which is less computationally-efficient).

5.2 Related Work

Sentence representations pre-trained from a large amount of data have been shown to be effective when transferred to a wide range of downstream tasks. Prior work along this line can be roughly divided into two categories: *i*) pre-trained models that require fine-tuning on the specific transferring task [DL15a, RH18, RNSS18, DCLT18, CYyK$^+$18]; *ii*) methods that extract general-purpose sentence embeddings, which can be effectively applied to downstream NLP tasks *without* fine-tuning the encoder parameters [KZS$^+$15a, HCK16, JBS17, GPH$^+$17, AKB$^+$17, LL18, PGJ18, TdS18]. Our proposed methods belong to the second category and provide a generic and easy-to-use encoder to extract highly informative sentence representations. However, our work is unique since the embeddings inferred from our models are binarized and compact, and thus possess the advantages of small memory footprint and much faster sentence retrieval.

Learning memory-efficient embeddings with deep neural networks has attracted substantial attention recently. One general strategy towards this goal is to extract discrete or binary data representations [JGP16, SN17, DGK$^+$17, CMS18, SSC$^+$18, THG19]. Binarized embeddings are especially attractive because they are more memory-efficient (relative to discrete embeddings), and they also enjoy the advantages of fast retrieval based upon a Hamming distance calculation. Previous work along this line in NLP has mainly focused on learning compact representations at the word-level [SN17, CMS18, THG19], while much less effort has been devoted to extracting binarized embeddings at the sentence-level. Our work aims to bridge this gap, and serves as an initial attempt to facilitate the deployment of state-of-the-art sentence embeddings on on-device mobile applications.

Our work is also related to prior research on semantic hashing, which aims to learn binary text embeddings specifically for the information retrieval task [SH09, ZWCL10, WSSJ14, XWT$^+$15, SSC$^+$18]. However, these methods are typically trained and evaluated on documents that belong to a specific domain, and thus cannot serve as generic binary sentence representation applicable to a wide variety of NLP taks. In contrast, our model is trained on large corpora and seeks to provide general-purpose binary representations that can be leveraged for various application scenarios.

5.3 Proposed Approach

We aim to produce compact and binarized representations from continuous sentence embeddings, and preserve the associated semantic information. Let x and f denote, respectively, an input sentence and the function defined by a pre-trained general-purpose sentence encoder. Thus, $f(x)$ represents the continuous embeddings extracted by the encoder. The goal of our model is to learn a universal transformation g that can convert $f(x)$ to highly informative binary sentence representations, *i.e.*, $g(f(x))$, which can be used as generic features for a collection of downstream tasks. We explore four strategies to parametrize the transformation g.

5.3.1 Hard Threshold

We use h and b to denote the continuous and binary sentence embeddings, respectively, and L denotes the dimension of h. The first method to binarize the continuous representations is to simply convert each dimension to either 0 or 1 based on a hard threshold. This strategy requires no training and directly operates on pre-trained continuous embeddings. Suppose s is the hard threshold, we have, for $i = 1, 2, ..., L$:

$$b^{(i)} = \mathbf{1}_{h^{(i)}>s} = \frac{\operatorname{sign}(h^{(i)} - s) + 1}{2}, \tag{5.1}$$

One potential issue of this direct binarization method is that the information contained in the continuous representations may be largely lost, since there is no training objective encouraging the preservation of semantic information in the produced binary codes [SSC+18]. Another disadvantage is that the length of the resulting binary code must be the same as the original continuous representation, and can not be flexibly chosen. In practice, however, we may want to learn shorter binary embeddings to save more memory footprint or computation.

5.3.2 Random Projection

To tackle the limitation of the above direct binarization method, we consider an alternative strategy that requires no training either: simply applying a random projection over the pre-trained continuous representations. [WK18] has shown that random sentence encoders can effectively construct universal sentence embeddings from word vectors, while possessing the flexibility of adaptively altering the embedding dimensions. Here, we are interested in exploring whether a random projection would also work well while transforming continuous sentence representations into their binary counterparts.

We randomly initialize a matrix $W \in \mathbb{R}^{D\times L}$, where D denotes the dimension of the resulting binary representations. Inspired by the standard initialization heuristic employed

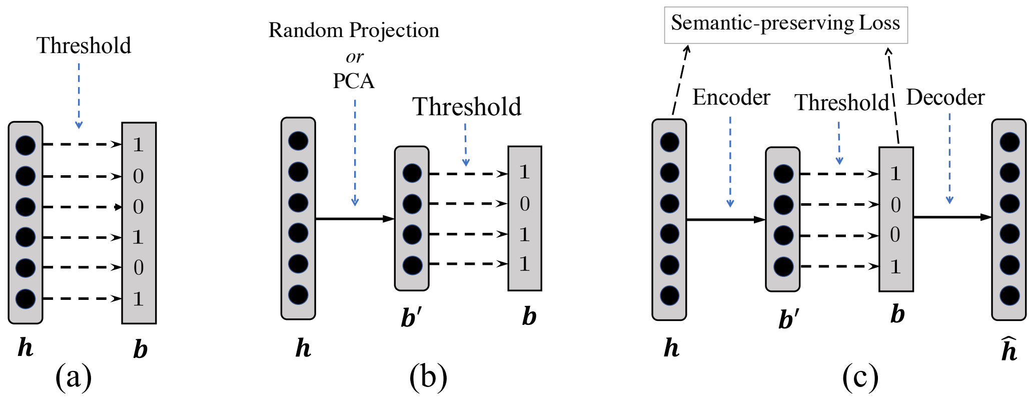

Figure 5.1: Proposed model architectures.

in [GB10, WK18], the values of the matrix are initialized as sampled uniformly. For $i = 1, 2, \ldots, D$ and $j = 1, 2, \ldots, L$, we have:

$$W_{i,j} \sim \text{Uniform}(-\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}}), \tag{5.2}$$

After converting the continuous sentence embeddings to the desired dimension D with the matrix randomly initialized above, we further apply the operation in (5.1) to binarize it into the discrete/compact form. The dimension D can be set arbitrarily with this approach, which is easily applicable to any pre-trained sentence embeddings (since no training is needed). This strategy is related to the Locality-Sensitive Hashing (LSH) for inferring binary embeddings [VDL10].

5.3.3 Principal Component Analysis

We also consider an alternative strategy to adaptively choose the dimension of the resulting binary representations. Specifically, Principal Component Analysis (PCA) is utilized to reduce the dimensionality of pre-trained continuous embeddings.

Given a set of sentences $\{x_i\}_{i=1}^N$ and their corresponding continuous embeddings $\{h_i\}_{i=1}^N \subset \mathbb{R}^L$, we learn a projection matrix to reduce the embedding dimensions while keeping the embeddings distinct as much as possible. After centralizing the embeddings as $h_i =$

$h_i - \frac{1}{N}\sum_{i=1}^{N} h_i$, the data, as a matrix $H = (h_1, h_2, \ldots, h_N)$, has the singular value decomposition (SVD):

$$H = U\Lambda V^T,$$

where Λ is an $L \times N$ matrix with descending singular values of X on its diagonal, with U and V orthogonal matrices. Then the correlation matrix can be written as: $HH^T = U\Lambda^2 U^T$. Assume that the diagonal matrix $\Lambda^2 = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_L)$ has descending elements $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L \geq 0$. We select first D rows of U as our projection matrix $W = U_{1:D}$, then the correlation matrix of WH is $WHH^TW^T = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$, which indicates that the embeddings are projected to D independent and most distinctive axes.

After projecting continuous embeddings to a representative lower dimensional space, we apply the hard threshold function at the position 0 to obtain the binary representations (since the embeddings are zero-centered).

5.3.4 Autoencoder Architecture

The methods proposed above suffer from the common issue that the model objective does not explicitly encourage the learned binary codes to retain the semantic information of the original continuous embeddings, and a separate binarization step is employed after training. To address this shortcoming, we further consider an autoencoder architecture, that leverages the reconstruction loss to hopefully endow the learned binary representations with more information. Specifically, an encoder network is utilized to transform the continuous into a binary latent vector, which is then reconstructed back with a decoder network.

For the encoder network, we use a matrix operation, followed by a binarization step, to extract useful features (similar to the random projection setup). Thus, for $i = 1, 2, \ldots, D$, we have:

$$b^{(i)} = \mathbf{1}_{\sigma(W_i \cdot h + k^{(i)}) > s^{(i)}} = \frac{\operatorname{sign}(\sigma(W_i \cdot h + k^{(i)}) - s^{(i)}) + 1}{2}, \tag{5.3}$$

where k is the bias term and $k^{(i)}$ corresponds to the i-th element of k. $s^{(i)}$ denotes the threshold determining whether the i-th bit is 0 or 1. During training, we may use either *deterministic* or *stochastic* binarization upon the latent variable. For the deterministic case, $s^{(i)} = 0.5$ for all dimensions; in the stochastic case, $s^{(i)}$ is uniformly sampled as: $s^{(i)} \sim \operatorname{Uniform}(0, 1)$. We conduct an empirical comparison between these two binarization strategies in Section 5.4.

Prior work have shown that linear decoders are favorable for learning binary codes under the encoder-decoder framework [CPR15, DGK+17, SSC+18]. Inspired by these results, we employ a linear transformation to reconstruct the original continuous embeddings from the binary codes:

$$\hat{h}^{(i)} = W_i' \cdot b + k'^{(i)}, \tag{5.4}$$

where W' and k' are weight and bias term respectively, which are learned. The mean square error between h and $\hat{h}$ is employed as the reconstruction loss:

$$\mathcal{L}_{rec} = \frac{1}{D} \sum_{i=1}^{D} (h^{(i)} - \hat{h}^{(i)})^2, \tag{5.5}$$

This objective imposes the binary vector b to encode more information from h (leading to smaller reconstruction error). Straight-through (ST) estimator [Hin12] is utilized to estimate the gradients for the binary variable. The autoencoder model is optimized by minimizing the reconstruction loss for all sentences. After training, the encoder network is leveraged as the transformation to convert the pre-trained continuous embeddings into the binary form.

Semantic-preserving Regularizer

Although the reconstruction objective can help the binary variable to endow with richer semantics, there is no loss that explicitly encourages the binary vectors to preserve the similarity information contained in the original continuous embeddings. Consequently, the model may lead to small reconstruction error but yield sub-optimal binary representations [THG19]. To improve the semantic-preserving property of the inferred binary embeddings, we introduce an additional objective term.

Consider a triple group of sentences $(x_\alpha, x_\beta, x_\gamma)$, whose continuous embeddings are $(h_\alpha, h_\beta, h_\gamma)$, respectively. Suppose that the cosine similarity between h_α and h_β is larger than that between h_β and h_γ, then it is desirable that the Hamming distance between b_α and b_β should be smaller than that between b_β and b_γ (notably, both large cosine similarity and small Hamming distance indicate that two sentences are semantically-similar).

Let $d_c(\cdot,\cdot)$ and $d_h(\cdot,\cdot)$ denote the cosine similarity and Hamming distance (in the continuous and binary embedding space), respectively. Define $l_{\alpha,\beta,\gamma}$ as an indicator such that, $l_{\alpha,\beta,\gamma} = 1$ if $d_c(h_\alpha, h_\beta) \geq d_c(h_\beta, h_\gamma)$, and $l_{\alpha,\beta,\gamma} = -1$ otherwise. The semantic-preserving regularizer is then defined as:

$$\mathcal{L}_{sp} = \sum_{\alpha,\beta,\gamma} \max\{0, l_{\alpha,\beta,\gamma}[d_h(b_\alpha, b_\beta) - d_h(b_\beta, b_\gamma)]\}, \tag{5.6}$$

By penalizing L_{sp}, the learned transformation function g is explicitly encouraged to retain the semantic similarity information of the original continuous embeddings. Thus, the entire objective function to be optimized is:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{sp}\mathcal{L}_{sp}, \tag{5.7}$$

where λ_{sp} controls the relative weight between the reconstruction loss ($\mathcal{L}_{rec}$) and semantic-preserving loss ($\mathcal{L}_{sp}$).

5.3.5 Discussion

Another possible strategy is to directly train the general-purpose binary embeddings from scratch, *i.e.*, jointly optimizing the continuous embeddings training objective and continuous-to-binary parameterization. However, our initial attempts demonstrate that this strategy leads to inferior empirical results. This observation is consistent with the results reported in [KC18]. Specifically, a binarization layer is directly appended over the InferSent architecture [CKS$^+$17] during training, which gives rise to much larger drop in terms of the embeddings' quality (we have conducted empirical comparisons with [KC18] in Table 5.1). Therefore, here we focus on learning universal binary embeddings based on pretained continuous sentence representations.

Model	**Dim**	**MR**	**CR**	**SUBJ**	**MPQA**	**SST**	**STS14**	**STSB**	**SICK-R**	**MRPC**
***Continuous** (**dense**) sentence embeddings*										
fastText-BoV	300	78.2	80.2	91.8	88.0	82.3	.65/.63	58.1/59.0	0.698	67.9/74.3
SkipThought	4800	76.5	80.1	93.6	87.1	82.0	.29/.35	41.0/41.7	0.595	57.9/66.6
SkipThought-LN	4800	79.4	83.1	**93.7**	89.3	82.9	.44/.45	-	-	-
InferSent-FF	4096	79.7	84.2	92.7	89.4	84.3	.68/.66	55.6/56.2	0.612	**67.9/73.8**
InferSent-G	4096	**81.1**	**86.3**	92.4	**90.2**	**84.6**	**.68/.65**	**70.0/68.0**	**0.719**	67.4/73.2
***Binary** (**compact**) sentence embeddings*										
InferLite-short	256	73.7	81.2	83.2	86.2	78.4	0.61/-	63.4/63.3	0.597	61.7/70.1
InferLite-medium	1024	76.3	83.2	87.8	88.4	81.3	0.67/-	64.9/64.9	0.642	64.1/72.0
InferLite-long	4096	77.7	83.7	89.6	89.1	82.3	0.68/-	67.9/67.6	0.663	65.4/**72.9**
HT-binary	4096	76.6	79.9	**91.0**	88.4	80.6	.62/.60	55.8/53.6	0.652	65.6/70.4
Rand-binary	2048	78.7	82.7	90.4	88.9	81.3	.66/.63	65.1/62.3	0.704	65.7/70.8
PCA-binary	2048	78.4	84.5	90.7	89.4	81.0	.66/.65	63.7/62.8	0.518	65.0/ 69.7
AE-binary	2048	78.7	**84.9**	90.6	89.6	82.1	.68/.66	71.7/69.7	0.673	65.8/70.8
AE-binary-SP	2048	**79.1**	84.6	90.8	**90.0**	**82.7**	**.69/.67**	**73.2/70.6**	**0.705**	**67.2**/72.0

Table 5.1: Performance on the test set for 10 downstream tasks.

5.4 Experimental setup

5.4.1 Pre-trained Continuous Embeddings

Our proposed model aims to produce highly informative binary sentence embeddings based upon pre-trained continuous representations. In this paper, we utilize InferSent [CKS$^+$17] as the continuous embeddings (given its effectiveness and widespread use). Note that all four proposed strategies can be easily extended to other pre-trained general-purpose sentence embeddings as well.

Specifically, a bidirectional LSTM architecture along with a max-pooling operation over the hidden units is employed as the sentence encoder, and the model parameters are optimized on the natural language inference tasks, *i.e.*, Standford Natural Language Inference (SNLI) [BAPM15c] and Multi-Genre Natural Language Inference (MultiNLI) datasets [WNB17].

5.4.2 Training Details

Our model is trained using Adam [KB14a], with a value 1×10^{-5} as the learning rate for all the parameters. The number of bits (*i.e.*, dimension) of the binary representation is set as 512, 1024, 2048 or 4096, and the best choice for each model is chosen on the validation set, and the corresponding test results are presented in Table 5.1. The batch size is chosen as 64 for all model variants. The hyperparameter over λ_{sp} is selected from $\{0.2, 0.5, 0.8, 1\}$ on the validation set, and 0.8 is found to deliver the best empirical results. The training with the autoencoder setup takes only about 1 hour to converge, and thus can be readily applicable to even larger datasets.

5.4.3 Evaluation

To facilitate comparisons with other baseline methods, we use SentEval toolkit[1] [CK18] to evaluate the learned binary (compact) sentence embeddings. Concretely, the learned representations are tested on a series of downstream tasks to assess their transferability (with the encoder weights fixed), which can be categorized as follows:

- **Sentence classification**, including sentiment analysis (MR, SST), product reviews (CR), subjectivity classification (SUBJ), opinion polarity detection (MPQA) and question type classification (TREC). A linear classifier is trained with the generic sentence embeddings as the input features. The default SentEval settings is used for all the datasets.

- **Sentence matching**, which comprises semantic relatedness (SICK-R, STS14, STSB) and paraphrase detection (MRPC). Particularly, each pair of sentences in STS14 dataset is associated with a similarity score from 0 to 5 (as the corresponding label). Hamming distance between the binary representations is directly leveraged as the prediction score (*without* any classifier parameters).

For the sentence matching benchmarks, to allow fair comparison with the continuous embeddings, we do not use the same classifier architecture in SentEval. Instead, we obtain the predicted relatedness by directly computing the *cosine similarity* between the continuous embeddings. Consequently, there are no classifier parameters for both the binary and continuous representations. The same valuation metrics in SentEval[CK18] are utilized for all the tasks. For MRPC, the predictions are made by simply judging whether a sentence pair's score is larger or smaller than the averaged Hamming distance (or cosine similarity).

[1]https://github.com/facebookresearch/SentEval

5.4.4 Baselines

We consider several strong baselines to compare with the proposed methods, which include both **continuous** (dense) and **binary** (compact) representations. For the continuous generic sentence embeddings, we make comparisons with fastText-BoV [JGBM16], Skip-Thought Vectors [KZS+15a] and InferSent [CKS+17]. As to the binary embeddings, we consider the binarized version of InferLite [KC18], which, as far as we are concerned, is the only general-purpose binary representations baseline reported.

5.5 Experimental Results

We experimented with five model variants to learn general-purpose binary embeddings: *HT-binary* (hard threshold, which is selected from $\{0, 0.01, 0.1\}$ on the validation set), *Rand-binary* (random projection), *PCA-binary* (reduce the dimensionality with principal component analysis), *AE-binary* (autoencoder with the reconstruction objective) and *AE-binary-SP* (autoencoder with both the reconstruction objective and Semantic-Preserving loss). Our code will be released to encourage future research.

5.5.1 Task transfer evaluation

We evalaute the binary sentence representations produced by different methods with a set of transferring tasks. The results are shown in Table 5.1. The proposed autoencoder architecture generally demonstrates the best results. Especially while combined with the semantic-preserving loss defined in (5.7), AE-binary-SP exhibits higher performance compared with a standard autoencoder. It is worth noting that the Rand-binary and PCA-binary model variants also show competitive performance despite their simplicity. These strategies are also quite promising given that no training is required given the pre-trained continuous sentence representations.

Another important result is that, the AE-binary-SP achieves competitive results relative to the InferSent, leading to only about 2% loss on most datasets and even performing at par with InferSent on several datasets, such as the MPQA and STS14 datasets. On the sentence matching tasks, the yielded binary codes are evaluated by merely utilizing the hamming distance features (as mentioned above). To allow fair comparison, we compare the predicted scores with the cosine similarity scores based upon the continuous representations (there are no additional parameters for the classifier). The binary codes brings out promising empirical results relative to their continuous counterparts, and even slightly outperform InferSent on the STS14 dataset.

We also found that our AE-binary-SP model variant consistently demonstrate superior results than the InferLite baselines, which optimize the NLI objective directly over the binary representations. This may be attributed to the difficulty of backpropagating gradients through discrete/binary variables, and would be an interesting direction for future research.

5.5.2 Nearest Neighbor Retrieval

Case Study One major advantage of binary sentence representations is that the similarity of two sentences can be evaluated by merely calculating the hamming distance between their binary codes. To gain more intuition regarding the semantic information encoded in the binary embeddings, we convert all the sentences in the SNLI dataset into continuous and binary vectors (with InferSent-G and AE-binary-SP, respectively). The top-3 closet sentences are retrieved based upon the corresponding metrics, and the resulting samples are shown in Table 5.2. It can be observed that the sentences selected based upon the Hamming distance indeed convey very similar semantic meanings. In some cases, the results with binary codes are even more reasonable compared with the continuous embeddings. For example, for the first query, all three sentences in the left column relate to "watching a movie", while one of the sentences in the right column is about "sleeping".

Hamming Distance (binary embeddings)	**Cosine Similarity (continuous embeddings)**
Query: Several people are sitting in a movie theater .	
A group of people watching a movie at a theater .	A group of people watching a movie at a theater .
A crowd of people are watching a movie indoors .	A man is watching a movie in a theater .
A man is watching a movie in a theater .	Some people are sleeping on a sofa in front of the television .
Query: A woman crossing a busy downtown street .	
A lady is walking down a busy street .	A woman walking on the street downtown .
A woman is on a crowded street .	A lady is walking down a busy street .
A woman walking on the street downtown .	A man and woman walking down a busy street .
Query: A well dressed man standing in front of piece of artwork .	
A well dressed man standing in front of an abstract fence painting .	A man wearing headphones is standing in front of a poster .
A man wearing headphones is standing in front of a poster .	A man standing in front of a chalkboard points at a drawing .
A man in a blue shirt standing in front of a garage-like structure painted with geometric designs .	A man in a blue shirt standing in front of a garage-like structure painted with geometric designs .
Query: A woman is sitting at a bar eating a hamburger .	
A woman sitting eating a sandwich .	A woman is sitting in a cafe eating lunch .
A woman is sitting in a cafe eating lunch .	A woman is eating at a diner .
The woman is eating a hotdog in the middle of her bedroom .	A woman is eating her meal at a resturant .
Query: Group of men trying to catch fish with a fishing net .	
Two men are on a boat trying to fish for food during a sunset .	There are three men on a fishing boat trying to catch bass .
There are three men on a fishing boat trying to catch bass .	Two men are trying to fish .
Two men pull a fishing net up into their red boat .	Two men are on a boat trying to fish for food during a sunset .

Table 5.2: Nearest neighbor retrieval results on the SNLI dataset.

Retrieval Speed The bitwise comparison is much faster than the element-wise multiplication operation (between real-valued vectors) [THG19]. To verify the speed improvement, we sample 10000 sentence pairs from SNLI and extract their continuous and binary embeddings (with the same dimension of 4096), respectively. We record the time to compute the cosine similarity and hamming distance between the corresponding representations. With our Python implementation, it takes 3.67μs and 288ns respectively, indicating that calculating the Hamming distance is over 12 times faster. Our implementation is not optimized, and the running time of computing Hamming distance can be further improved (to be proportional to the number of different bits, rather than the input length[2]).

5.5.3 Ablation Study

The effect of semantic-preserving loss

To investigate the importance of incorporating the locality-sensitive regularizer, we select different values of λ_{sp} (ranging from 0.0 to 1.0) and explore how the transfer results would change accordingly. The λ_{sp} controls the relative weight of the semantic-preserving loss term. As shown in Table 5.3, augmenting the semantic-preserving loss consistently improves the quality of learned binary embeddings, while the best test accuracy on the MR dataset is obtained with $\lambda_{sp} = 0.8$.

$\boldsymbol{\lambda}_{sp}$	0.0	0.2	0.5	0.8	1.0
Accuracy	78.2	78.5	78.5	**79.1**	78.4

Table 5.3: Ablation study for the AE-binary-SP model with different choices of λ_{sp} (evaluated with test accuracy on the MR dataset).

[2]https://en.wikipedia.org/wiki/Hamming_distance

Sampling strategy

As discussed in Section 5.3.4, the binary latent vector b can be obtained with either a deterministic or stochastically sampled threshold. We compare these two sampling strategies on several downstream tasks. As illustrated in Figure 5.2, setting a fixed threshold demonstrates better empirical performance on all the datasets. Therefore, deterministic threshold is employed for all the autoencoder model variants in our experiments.

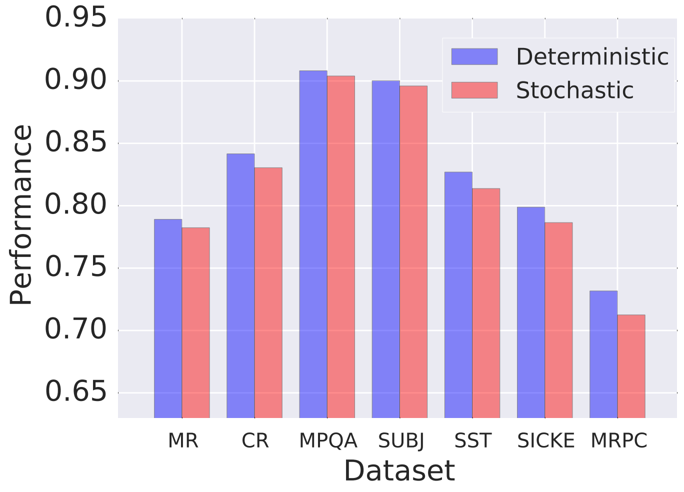

Figure 5.2: The comparison between *deterministic* and *stochastic* sampling for the autoencoder strategy.

The effect of embedding dimension

Except for the hard threshold method, other three proposed strategies all possess the flexibility of adaptively choosing the dimension of learned binary representations. To explore the sensitivity of extracted binary embeddings to their dimensions, we run four model variants (Rand-binary, PCA-binary, AE-binary, AE-binary-SP) with different number of bits (*i.e.*, 512, 1024, 2048, 4096), and their corresponding results on the MR dataset are shown in Figure 5.3.

For the AE-binary and AE-binary-SP models, longer binary codes consistently deliver

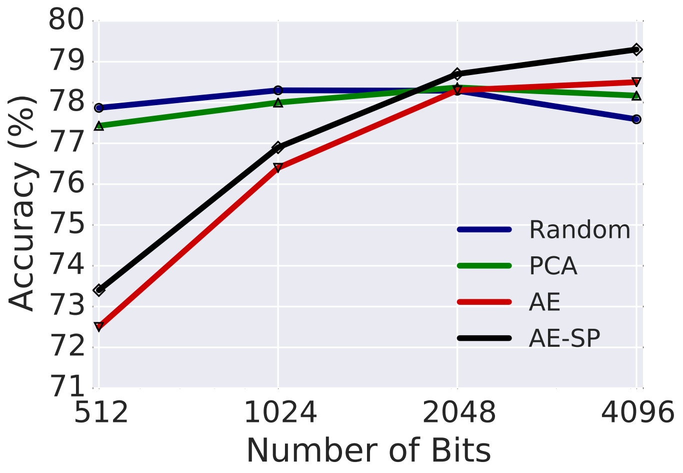

Figure 5.3: The test accuracy of different model on the MR dataset across 512, 1024, 2048, 4096 bits for the learned binary representations.

better results. While for the Rand-binary and PCA-binary variants, the quality of inferred representations is much less sensitive to the embedding dimension. Notably, these two strategies exhibit competitive performance even with only 512 bits. Therefore, in the case where less memory footprint or little training is preferred, Rand-binary and PCA-binary could be more judicious choices.

5.6 Conclusion

This paper presents a first step towards learning binary and general-purpose sentence representations that allow for efficient storage and fast retrieval over massive corpora. To this end, we explore four distinct strategies to convert pre-trained continuous sentence embeddings into a binarized form. Notably, a regularized autoencoder augmented with semantic-preserving loss exhibits the best empirical results, degrading performance by only around 2% while saving over 98% memory footprint. Besides, two other model variants with a random projection or PCA transformation require no training and demonstrate competitive embedding quality even with relatively small dimensions. Experiments on nearest-neighbor sentence retrieval further validate the effectiveness of proposed framework.

Chapter 6

Conclusions

Deep latent-variable models have seen amazing advancements in recent years, and will continue to help pushing the field of natural language processing (NLP). In this dissertation, I mainly discuss the research contribution I have made on this direction, which can be summarized as follows:

In Chapter 2, I presented a latent-variable model with a deconvolutional sequence decoder, which is optimized by jointly optimizing the variational lower bound and a matching loss. It is shown that the model effectively extracts semantically-meaningful representations, even with limited amount of labeled data available. We evaluate the proposed method in cases of both unsupervised an supervised learning. Specifically, the deconvolutional decoder is demonstrated to be more beneficial than its LSTM counterpart when it comes to obtaining sentence embeddings.

In Chapter 3, I introduced an end-to-end variational autoencoder architecture for semantic hashing, where binary sentence representations can be learned in an unsupervised manner. Specifically, a Bernoulli prior is assumed on the latent variable, and the gradients can be backpropogated through it thanks to the straight-through estimator. Moreover, we propose to inject data-dependent noise into the Bernoulli latent variable during training, inspired by the rate-distortion theory.

In Chapter 4, I discussed my research efforts towards facilitating long-form text generation. I hierarchically-structured variational autoencoder is proposed to model the word- and sentence-level semantic coherence simultaneously. Besides, to alleviate the posterior collapse issue inherent in variational autoencoder models for text, I proposed to employ a hierarchy of stochastic layers between the inference and generative networks. As a result, the latent representations learned from our model are semantically-richer relative to several strong baselines. More importantly, it is demonstrated, with both automatic metrics and human evaluation, that the generated samples are with superior quality.

In Chapter 4, the general idea of learning binary embeddings is extended to the case of learning general-purpose sentence representations, which enjoys the advantages of: 1) faster retrieval speed over massive corpora; 2) more storage-efficient compared with the continuous counterpart. Four different strategies are explored to convert pre-trained continuous embeddings into a binarized form. Notably, a regularized autoencoder augmented with semantic-preserving loss exhibits the best empirical results.

In terms of future works, deep latent-variable models could be useful for many other challenging NLP problems, such as semi-supervised learning, transfer learning, model compression, data augmentation, *etc*.

Bibliography

[AAB$^+$16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[ABC$^+$14] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91, 2014.

[ABC$^+$16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[ACB17] Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[AKB$^+$17] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207, 2017.

[Ame83] American Psychological Association. *Publications Manual*. American Psychological Association, Washington, DC, 1983.

[Ass83] Association for Computing Machinery. *Computing Reviews*, 24(11):503–512, 1983.

[AU72] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[BAPM15a] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *EMNLP*, 2015.

[BAPM15b] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[BAPM15c] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. in international conference on learning representations. *ICLR*, 2015.

[Ber03] Toby Berger. Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*, 2003.

[BGL$^+$94] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[BGR$^+$16] Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016.

[BGWB14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[BL07] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[BLS16] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *2016 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2016.

[BMVP18] Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. Variational attention for sequence-to-sequence models. In *COLING*, 2018.

[BVV$^+$15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[BVV$^+$16a] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoNLL 2016*, page 10, 2016.

[BVV$^+$16b] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL*, 2016.

[CAR16] Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. *NAACL*, 2016.

[CBS18] Asli Celikyilmaz, Antoine Bosselut, and Dinghan Shen. Arxgen: Corpus of arxiv articles for deep generative models. *https://github.com/Microsoft/ARXGEN/tree/master/arxiv*, 2018.

[CBT$^+$] Liqun Chen, Ke Bai, Chenyang Tao, Yizhe Zhang, Guoyin Wang, Wenlin Wang, Ricardo Henao, and Lawrence Carin. Sequence generation with optimal-transport-enhanced reinforcement learning.

[CDP$^+$17] Liqun Chen, Shuyang Dai, Yunchen Pu, Chunyuan Li, Qinliang Su, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. *arXiv preprint arXiv:1709.01846*, 2017.

[CDT$^+$18] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4671–4682, 2018.

[CF17] Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *Proceedings of the 40th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2017.

[CK18] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[CKS$^+$16] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[CKS$^+$17] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, 2017.

[CLL$^+$19] Pengyu Cheng, Chang Liu, Chunyuan Li, Dinghan Shen, Ricardo Henao, and Lawrence Carin. Straight-through estimator as projected wasserstein gradient flow. *arXiv preprint arXiv:1910.02176*, 2019.

[CLZ$^+$] Pengyu Cheng, Yitong Li, Xinyuan Zhang, Liqun Chen, David Carlson, and Lawrence Carin. Dynamic embedding on textual networks via a gaussian process.

[CLZ^{+}19] Pengyu Cheng, Yitong Li, Xinyuan Zhang, Liqun Cheng, David Carlson, and Lawrence Carin. Gaussian-process-based dynamic embedding for textual networks. *arXiv preprint arXiv:1910.02187*, 2019.

[CMS18] Ting Chen, Martin Renqiang Min, and Yizhou Sun. Learning k-way d-dimensional discrete codes for compact embedding representations. *arXiv preprint arXiv:1806.09464*, 2018.

[CPR15] Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 557–566. IEEE, 2015.

[CSAF18] Ondřej Cífka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*, 2018.

[CTLG18] Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. Variational sequential labelers for semi-supervised learning. In *EMNLP*, 2018.

[CTZ^{+}18] Liqun Chen, Chenyang Tao, Ruiyi Zhang, Ricardo Henao, and Lawrence Carin Duke. Variational inference and model selection with generalized evidence bounds. In *International Conference on Machine Learning*, pages 892–901, 2018.

[CYyK^{+}18] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[CZZ^{+}19] Liqun Chen, Yizhe Zhang, Ruiyi Zhang, Chenyang Tao, Zhe Gan, Haichao Zhang, Bai Li, Dinghan Shen, Changyou Chen, and Lawrence Carin. Improving sequence-to-sequence learning via optimal transport. *arXiv preprint arXiv:1901.06283*, 2019.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DGK^{+}17] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. *arXiv preprint arXiv:1701.02815*, 2017.

[DGM06] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer, 2006.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab 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.

[DKC$^+$18] Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. Latent alignment and variational attention. *CoRR*, abs/1807.03756, 2018.

[DL15a] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[DL15b] Andrew M Dai and Quoc V Le. Semi-supervised Sequence Learning. *NIPS*, 2015.

[DLH$^+$18] Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. Variational autoregressive decoder for neural response generation. In *EMNLP*, 2018.

[FGD18] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

[FLD18] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *ACL*, 2018.

[FLL$^+$19] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigate kl vanishing. *NAACL*, 2019.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[GCHK18] Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. Dialogwae: Multimodal response generation with conditional wasserstein autoencoder. *arXiv preprint arXiv:1805.12352*, 2018.

[GCW$^+$17] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5247–5256, 2017.

[GDG$^+$15] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[GGH+17] Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5630–5639, 2017.

[GHOL18] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *TACL*, 6:437–450, 2018.

[GKA+16] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

[GLC+16] Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034*, 2016.

[GPH+16] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning generic sentence representations using convolutional neural networks. *arXiv preprint arXiv:1611.07897*, 2016.

[GPH+17] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning generic sentence representations using convolutional neural networks. In *EMNLP*, 2017.

[GSC+17] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *NIPS*, 2017.

[HBCW15] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[HBF+18] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. *ACL*, 2018.

[HCK16] Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*, 2016.

[HCS+16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[HGOL18] Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062, 2018.

[Hin12] Geoffrey Hinton. Neural networks for machine learning, coursera. *URL: http://coursera. org/course/neuralnets*, 2012.

[HLLC14a] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[HLLC14b] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional Neural Network Architectures for Matching Natural Language Sentences. *NIPS*, 2014.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSNB19] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *ICLR*, 2019.

[HYL$^+$17] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *ICML*, 2017.

[HZG17] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

[IAMB17] Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising Criterion for Variational Auto-Encoding Framework. *AAAI*, 2017.

[JBS17] Yacine Jernite, Samuel R. Bowman, and David A Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR*, abs/1705.00557, 2017.

[JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[KB14a] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB14b] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KC18] Jamie Kiros and William Chan. Inferlite: Simple universal sentence representations from natural language inference data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4868–4874, 2018.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[KMRW14a] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[KMRW14b] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. *NIPS*, 2014.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[KRV$^+$18] Lukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *ICML*, 2018.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KWM$^+$18] Yoon Kim, Sam Wiseman, Andrew C. Miller, David A Sontag, and Alexander M. Rush. Semi-amortized variational autoencoders. In *ICML*, 2018.

[KZR$^+$17] Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*, 2017.

[KZS$^+$15a] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015.

[KZS$^+$15b] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[LGC$^+$19] Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, Lawrence Carin, et al. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

[LGS$^+$19] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. Storygan: A sequential conditional gan for story visualization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6329–6338, 2019.

[LL12] Hugo Larochelle and Stanislas Lauly. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716, 2012.

[LL18] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *ICLR*, 2018.

[LLC$^+$17] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*, pages 5495–5503, 2017.

[LLJ15a] Jiwei Li, Minh-Thang Luong, and Daniel Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*, 2015.

[LLJ15b] Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*, volume 1, pages 1106–1115, 2015.

[LLZ$^+$] Yuan Li, Chunyuan Li, Yizhe Zhang, Xiujun Li, Guoqing Zheng, Lawrence Carin, and Jianfeng Gao. Complementary auxiliary classifiers for label-conditional text generation.

[LMS$^+$18] Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[LSDJ06] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh 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.

[LSHT17] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. *arXiv preprint arXiv:1705.10999*, 2017.

[LWJ$^+$12] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE, 2012.

[LWL$^+$19] Kevin Liang, Guoyin Wang, Yitong Li, Ricardo Henao, and Lawrence Carin. Kernel-based approaches for sequence modeling: Connections to neural methods. In *Advances in Neural Information Processing Systems*, pages 3387–3398, 2019.

[MGB17a] Yishu Miao, Edward Grefenstette, and Phil Blunsom. Discovering discrete latent topics with neural variational inference. In *ICML*, 2017.

[MGB17b] Yishu Miao, Edward Grefenstette, and Phil Blunsom. Discovering discrete latent topics with neural variational inference. *arXiv preprint arXiv:1706.00359*, 2017.

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[MKB$^+$11] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE, 2011.

[MML$^+$15] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*, 2015.

[MML$^+$16] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang 0023, Rui Yan, and Zhi Jin. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. *ACL*, 2016.

[MMT16] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[MPH08] CD Manning, R PRABHAKAR, and S HINRICH. Introduction to information retrieval, volume 1 cambridge university press. *Cambridge, UK*, 2008.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[MSJG15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoder. *CoRR*, 2015.

[MYB16a] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *Proc. ICML*, 2016.

[MYB16b] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *ICML*, 2016.

[NBG17] Allen Nie, Erin D. Bennett, and Noah D. Goodman. Dissent: Sentence representation learning from explicit discourse relations. *CoRR*, abs/1710.04334, 2017.

[PBC$^+$09] Sandeep Pandey, Andrei Broder, Flavio Chierichetti, Vanja Josifovski, Ravi Kumar, and Sergei Vassilvitskii. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th international conference on World wide web*, pages 441–450. ACM, 2009.

[PCK18] Yookoon Park, Jaemin Cho, and Gunhee Kim. A hierarchical latent structure for variational conversation modeling. In *NAACL-HLT*, 2018.

[PDG$^+$18] Yunchen Pu, Shuyang Dai, Zhe Gan, Weiyao Wang, Guoyin Wang, Yizhe Zhang, Ricardo Henao, and Lawrence Carin. Jointgan: Multi-domain joint distribution learning with generative adversarial nets. *arXiv preprint arXiv:1806.02978*, 2018.

[PGH$^+$16a] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems*, pages 2352–2360, 2016.

[PGH$^+$16b] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational Autoencoder for Deep Learning of Images, Labels and Captions. *NIPS*, 2016.

[PGH$^+$17] Yuchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. Vae learning via stein variational gradient descent. In *Advances in Neural Information Processing Systems*, pages 4236–4245, 2017.

[PGJ18] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*, 2018.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[PWH$^+$17] Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 4330–4339, 2017.

[RBAD14] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

[RCAZ15] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[RER$^+$18a] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[RER$^+$18b] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *ICML*, 2018.

[RGH$^+$15] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[RH18] Sebastian Ruder and Jeremy Howard. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[RK18] Sujith Ravi and Zornitsa Kozareva. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 804–810, 2018.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[SB18] Harshil Shah and David Barber. Generative neural machine translation. *arXiv preprint arXiv:1806.05138*, 2018.

[SCS$^+$19] Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, Asli Celikyilmaz, and Lawrence Carin. Learning compressed sentence representations for on-device text processing. *arXiv preprint arXiv:1906.08340*, 2019.

[SCZ$^+$19] Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, and Lawrence Carin. Towards generating long and coherent text with multi-level latent variable models. *arXiv preprint arXiv:1902.00154*, 2019.

[SH09] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[SHK$^+$14a] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SHK$^+$14b] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SHP$^+$11] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809, 2011.

[SKS$^+$17] Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Christopher Joseph Pal, and Yoshua Bengio. Twin networks: Using the future as a regularizer. *CoRR*, abs/1708.06742, 2017.

[SMLC17a] Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. Adaptive convolutional filter generation for natural language understanding. *arXiv preprint arXiv:1709.08294*, 2017.

[SMLC17b] Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. Learning context-sensitive convolutional filters for text processing. *arXiv preprint arXiv:1709.08294*, 2017.

[SN17] Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. *arXiv preprint arXiv:1711.01068*, 2017.

[SRM$^+$16] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[SSB17a] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A Hybrid Convolutional Variational Autoencoder for Text Generation. *arXiv.org*, February 2017.

[SSB17b] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *EMNLP*, 2017.

[SSC$^+$18] Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *ACL*, 2018.

[SSL$^+$17a] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, 2017.

[SSL$^+$17b] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. *AAAI*, 2017.

[SSL$^+$17c] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301, 2017.

[SSL$^+$17d] Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. A Conditional Variational Framework for Dialog Generation. *arXiv.org*, April 2017.

[SSL$^+$17e] Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. A conditional variational framework for dialog generation. In *ACL*, 2017.

[SSL$^+$17f] Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. A conditional variational framework for dialog generation. *arXiv preprint arXiv:1705.00316*, 2017.

[SSW$^+$19] Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*, 2019.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[SWW$^+$18] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*, 2018.

[SzEP07] Benno Stein, Sven Meyer zu Eissen, and Martin Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 825–826. ACM, 2007.

[SZH$^+$18a] Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. Deconvolutional latent-variable model for text sequence matching. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[SZH$^+$18b] Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. Deconvolutional latent-variable model for text sequence matching. In *AAAI*, 2018.

[SZHC18] Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Improved semantic-aware network embedding with fine-grained word alignment. *arXiv preprint arXiv:1808.09633*, 2018.

[TCH$^+$18] Chenyang Tao, Liqun Chen, Ricardo Henao, Jianfeng Feng, and Lawrence Carin Duke. Chi-square generative adversarial network. In *International Conference on Machine Learning*, pages 4894–4903, 2018.

[TDC$^+$19] Chenyang Tao, Shuyang Dai, Liqun Chen, Ke Bai, Junya Chen, Chang Liu, Ruiyi Zhang, Georgiy Bobashev, and Lawrence Carin Duke. Variational annealing of gans: A langevin perspective. In *International Conference on Machine Learning*, pages 6176–6185, 2019.

[TDLL18] Trieu H Trinh, Andrew M Dai, Thang Luong, and Quoc V Le. Learning longer-term dependencies in rnns with auxiliary losses. 2018.

[TdS18] Shuai Tang and Virginia R de Sa. Improving sentence representations with multi-view frameworks. *arXiv preprint arXiv:1810.01064*, 2018.

[THG19] Julien Tissier, Amaury Habrard, and Christophe Gravier. Near-lossless binarization of word embeddings. *AAAI*, 2019.

[TMM$^+$17] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2624–2633, 2017.

[TSCH17] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *ICLR*, 2017.

[UJ17] Alexander Schwing Unnat Jain, Ziyu Zhang. Creativity: Generating diverse questions using variational autoencoders. *CVPR*, 2017.

[VDL10] Benjamin Van Durme and Ashwin Lall. Online generation of locality sensitive hash signatures. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 231–235. Association for Computational Linguistics, 2010.

[vdOV$^+$17] Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6309–6318, 2017.

[WBGL16] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2016.

[WG18] John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, 2018.

[WGW$^+$17] Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*, 2017.

[WGW$^+$18] Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. Topic compositional neural language model. In *AISTATS*, 2018.

[WGX$^+$19a] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational autoencoders for text generation. *arXiv preprint arXiv:1903.07137*, 2019.

[WGX$^+$19b] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Yingxu Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational autoencoders for text generation. *NAACL*, 2019.

[WHF17] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. *CoRR*, 2017.

[WJ16] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[WK18] John Wieting and Douwe Kiela. No training required: Exploring random encoders for sentence classification. *CoRR*, abs/1901.10444, 2018.

[WKC10] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

[WKC12] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

[WLW$^+$18] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[WNB17] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[WPV$^+$18] Wenlin Wang, Yunchen Pu, Vinay Kumar Verma, Kai Fan, Yizhe Zhang, Changyou Chen, Piyush Rai, and Lawrence Carin. Zero-shot learning via class-conditioned deep generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[WSSJ14] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[WTF09] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[WTG$^+$19] Wenlin Wang, Chenyang Tao, Zhe Gan, Guoyin Wang, Liqun Chen, Xinyuan Zhang, Ruiyi Zhang, Qian Yang, Ricardo Henao, and Lawrence Carin. Improving textual network learning with variational homophilic embeddings. In *Advances in Neural Information Processing Systems*, pages 2074–2085, 2019.

[WXG$^+$19] Wenlin Wang, Hongteng Xu, Zhe Gan, Bai Li, Guoyin Wang, Liqun Chen, Qian Yang, Wenqi Wang, and Lawrence Carin. Graph-driven generative models for heterogeneous multi-task learning. *arXiv preprint arXiv:1911.08709*, 2019.

[WXW$^+$19] Wenlin Wang, Hongteng Xu, Guoyin Wang, Wenqi Wang, and Lawrence Carin. An optimal transport framework for zero-shot learning. *arXiv preprint arXiv:1910.09057*, 2019.

[WXZ$^+$19] Wenlin Wang, Hongteng Xu, Ruiyi Zhang, Wenqi Wang, and Lawrence Carin. Collaborative filtering with a synthetic feedback loop. *arXiv preprint arXiv:1910.12735*, 2019.

[WZS13] Qifan Wang, Dan Zhang, and Luo 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.

[XD18] Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. In *EMNLP*, 2018.

[XSDT17a] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, pages 3358–3364, 2017.

[XSDT17b] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational Autoencoder for Semi-Supervised Text Classification. *AAAI*, 2017.

[XWLC18] Hongteng Xu, Wenlin Wang, Wei Liu, and Lawrence Carin. Distilled wasserstein learning for word embedding and topic modeling. In *Advances in Neural Information Processing Systems*, pages 1716–1725, 2018.

[XWT$^+$15] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. Convolutional neural networks for text hashing. In *IJCAI*, pages 1369–1375, 2015.

[YHBP14] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[YHSBK17a] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*, 2017.

[YHSBK17b] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *ICML*, 2017.

[YHWC19] Qian Yang, Zhouyuan Huo, Wenlin Wang, and Lawrence Carin. Ouroboros: On accelerating training of transformer-based language models. In *Advances in Neural Information Processing Systems*, pages 5520–5530, 2019.

[YSC$^+$19] Qian Yang, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang, Lawrence Carin, et al. An end-to-end generative architecture for paraphrase generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3123–3133, 2019.

[YWH$^+$16] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*, pages 4584–4593, 2016.

[YYD$^+$16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In

Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489, 2016.

[YZHN18] Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *ACL*, 2018.

[YZWY17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[ZGF$^+$17a] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *ICML*, 2017.

[ZGF$^+$17b] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.

[ZGG$^+$18] Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. Generating informative and diverse conversational responses via adversarial information maximization. In *NIPS*, 2018.

[ZHG$^+$18] Xinyuan Zhang, Ricardo Henao, Zhe Gan, Yitong Li, and Lawrence Carin. Multi-label learning from medical plain text with convolutional residual models. *arXiv preprint arXiv:1801.05062*, 2018.

[ZKTF10] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[ZKZ$^+$15] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27, 2015.

[ZKZ$^+$18] Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders. In *ICML*, 2018.

[ZLdM18] Xunjie Zhu, Tingfeng Li, and Gerard de Melo. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting* *of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 632–637, 2018.

[ZLE18] Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*, 2018.

[ZLSC18] Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. Diffusion maps for textual network embedding. In *Advances in Neural Information Processing Systems*, pages 7587–7597, 2018.

[ZLZ$^+$18] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*, 2018.

[ZSW$^+$17] Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179, 2017.

[ZWCL10] Dell Zhang, Jun Wang, Deng Cai, and Jinsong 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.

[ZXS$^+$16a] Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*, 2016.

[ZXS$^+$16b] Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. Variational neural machine translation. In *EMNLP*, 2016.

[ZYS$^+$19] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. Text-based interactive recommendation via constraint-augmented reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15188–15198, 2019.

[ZYY$^+$19] Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. Syntax-infused variational autoencoder for text generation. *arXiv preprint arXiv:1906.02181*, 2019.

[ZZE17] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, 2017.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, 2015.

Biography

Dinghan Shen is currently a Ph.D. candidate of the Electrical and Computer Engineering Department at Duke University, advised by Prof. Lawrence Carin. His research interests are focused on the intersection of deep learning and natural language processing, and he is particularly interested in: 1) natural language reasoning and understanding tasks; (2) making text generation more scalable and reliable. He had over 20 papers published on top-tier conferences. Particularly, he received the Best Paper Honorable Mention award at ACL 2018 as the first author, and is also a co-recipient of the Best Student Paper at CVPR 2019. During the Ph.D. period, He had internship experience at Microsoft Research and Google AI. Before coming to Duke, he earned a Bachelor of Engineering degree from Peking University in 2015.