Discretized Bottleneck: Posterior-Collapse-Free Sequence-to-Sequence Learning

Yang Zhao[1], **Ping Yu**[1], **Suchismit Mahapatra**[2], **Qinliang Su**[3], **Changyou Chen**[1]
[1]University at Buffalo, [2]Criteo, [3]Duke Univeristy
{yzhao63, pingyu, changyou, suchismi}@buffalo.edu
qinliang.su@duke.edu

Abstract

Variational autoencoders (VAEs) are important tools in end-to-end representation learning. VAEs can capture complex data distributions and have been applied extensively in many natural-language-processing (NLP) tasks. However, a common pitfall in sequence-to-sequence learning with VAEs is the posterior-collapse issue in latent space, wherein the model tends to ignore latent variables when a strong auto-regressive decoder is implemented. In this paper, we propose a principled approach to eliminate this issue by applying a discretized bottleneck in the latent space. Specifically, we impose a shared discrete latent space where each input is learned to choose a combination of shared latent atoms as its latent representation. Compared with VAEs employing continuous latent variables, our model endows more promising capability in modeling underlying semantics of discrete sequences and can thus provide more interpretative latent structures. Empirically, we demonstrate the efficiency and effectiveness of our model on a broad range of tasks, including language modeling, unaligned text style transfer, dialog response generation, and neural machine translation.

1 Introduction

Auto-encoder models are widely used in various NLP tasks such as machine translation (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014; Luong et al., 2015; Vaswani et al., 2017; Devlin et al., 2018) and dialog response generation (Vinyals and Le, 2015; Olabiyi and Mueller, 2019) tasks. Generally speaking, an auto-encoder model learns a function to map each input to a latent representation and then back to the original data space.

Unlike Auto-encoders, VAEs aim to learn a probability distribution of a dataset, which can generate new instances that look similar to the original dataset. With such a generative model, one can easily draw samples from the distribution following a decoding scheme. VAEs have achieved tremendous success in generating high-quality images, videos, and speech (van den Oord et al., 2017; Razavi et al., 2019). At the same time, VAEs have also been applied in NLP to improve traditional maximum-likelihood-estimation (MLE) based models, achieving impressive progress in language modeling (Bowman et al., 2015; Fabius and van Amersfoort, 2014; Miao et al., 2016; Yang et al., 2017), controllable text generation (Hu et al., 2017), neural machine translation (Shah and Barber, 2018), and many other applications.

Although with impressive success, a well-known pitfall with VAEs, especially in applications of sequence-to-sequence (Seq2Seq) modeling, is a phenomenon called *latent variable collapse* (or posterior collapse) (Bowman et al., 2015), where an encoder yields meaningless posteriors that collapse to the prior. With this pitfall, VAEs usually fail to learn meaningful representations of individual data samples. Several attempts have been made to alleviate this issue (Bowman et al., 2015; Hoffman and Johnson, 2016; Sønderby et al., 2016; Kingma et al., 2016; Chen et al., 2016; Zhao et al., 2017b; Yeung et al., 2017; Alemi et al., 2017; Dieng et al., 2018; Fu et al., 2019; He et al., 2019; Fang et al., 2019), however most of these approaches are heuristic in nature.

Our solution is motivated by two possible explanations of posterior collapse: *i*) Recent research shows that the prior plays an important role in density estimation (Hoffman and Johnson, 2016; Takahashi et al., 2019). Although Gaussian prior and posterior are largely adopted, such simplified priors tend to incur *latent variable collapse* for poor density estimations. To overcome this issue, we argue that a flexible prior should be learned simul-

taneously during training. In this way, even if one encounters posterior collapse while learning, the collapsed variational distribution is still meaningful. *ii*) Related work has also shown that the posterior collapse is caused by a lack of good latent codes (Fu et al., 2019). Thus, designing an effective way of learning useful presentations without supervision is the key to address the problem. In this paper, based on the above two arguments, we propose to enforce a discrete latent space for VAEs. The discrete space consists of learnable atoms that are shared by all data inputs. The discrete latent space automatically brings in at least three benefits: *i*) The atoms of a discrete prior could be efficiently learned during training; *ii*) The discrete nature of a prior makes the KL-divergence between the prior and a variational distribution un-vanishable, thus free of posterior collapse; *iii*) The discrete VAE is formulated following the standard VAE setting, making learning and inference particularly efficient. The contributions of our paper are summarized as follows:

- We propose the concept of discretized bottleneck VAEs for RNN-based Seq2Seq models, which can overcome the posterior-collapse problem, a long-standing issue that needs to be well addressed in NLP applications.
- We showcase how to inject the discretized bottleneck in Seq2Seq models on a variety of NLP tasks. When a model and the training strategy are carefully managed, our DB-VAE can accurately model discrete text without scarifying reliance on latent representations and experiencing posterior collapse. We also find that under our framework, the discrete bottleneck can capture more sentence-level semantic features.
- Inference of the proposed DB-VAE requires a nearest-neighbor (NN) search for the discrete atoms in a latent space. We extend NN to the k-NN setting and show that it can provide more corrected translations given one source text, thus increase the BLEU score. The method is referred to as *top-k* search. Naturally, it can also provide diverse responses in the dialog response generation task.

2 Preliminaries

2.1 Variational Autoencoder

VAEs consist of two parts, an encoder (inference network) and a decoder (generative network). The decoder corresponds to the following generative process for an input $\mathbf{x}$:

$$\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}) \tag{1}$$

where $p(\mathbf{z})$ is a pre-defined prior distribution and $p_\theta(\mathbf{x}|\mathbf{z})$ is a conditional distribution (likelihood) induced by a decoder. To learn the parameters θ, one typically maximizes the following marginal log-likelihood:

$$\log p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} \tag{2}$$

Direct optimization of the log-likelihood is usually intractable. VAEs instead parameterize a family of variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ (often known as an encoder) to approximate the true posterior $p_\theta(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, ending up optimizing the following evidence lower bound (ELBO):

$$\begin{aligned}\log p_\theta(\mathbf{x}) \geqslant \text{ELBO} = E_{p_\theta(\mathbf{x})}\{E_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) \\ - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\}\end{aligned} \tag{3}$$

2.2 Posterior collapse

In neural language models, both the encoder and the decoder are often parameterized by strong autoregressive neural networks, *i.e.*, LSTM and GRU with an input $\mathbf{x} = \{x_1, ..., x_t, ..., x_T\}$, where every token x_t is fully conditioned on all previous tokens:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{t=0}^{T} p(x_t|x_{<t}, \mathbf{z}) \tag{4}$$

An issue with VAE is that $p_\theta(\mathbf{x}|\mathbf{z})$ is defined in a very flexible manner that allows pushing the KL term towards zero, leading to posterior collapse that learns meaningless latent codes (Bowman et al., 2015; Kingma et al., 2016; Chen et al., 2016).

3 Discretized Bottleneck in VAE

3.1 Model

Our proposed model is general and can be applied to most existing Seq2Seq models. Without loss of generality, we will describe our framework under the setting of an RNN-based language model. As shown in Figure 1, our model consists of three parts, an encoder, a latent code generator, and a decoder.

Encoder Let an input sequence be defined as $\mathbf{x} = \{x_1, ..., x_t, ..., x_T\}$. The encoder aims at encoding an input token at each time step to a latent representation. This is implemented by feeding an input sequence to an LSTM encoder, resulting in

$$\mathbf{h}_t^e = \textbf{LSTM}(\mathbf{w}_{\mathbf{x}_t}, \mathbf{h}_{t-1}^e) \,, \tag{5}$$

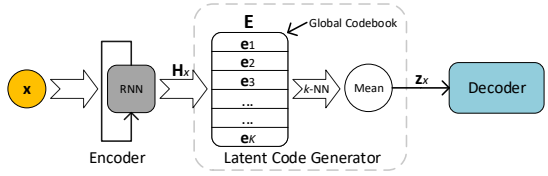

Figure 1: The graphical illustration of the proposed model

where $\mathbf{w}_{x_t}$ is the word embedding vector of the word x_t. The latent representation of the input sequence $\mathbf{x}$ is $\mathbf{H}_x \triangleq (\mathbf{h}_1^e, \cdots, \mathbf{h}_T^e)$.

Latent code generation Different from the vanilla VAE mechanism, we define a latent code to be a combination of a set of latent codes from a global codebook $\mathbf{E} \triangleq [\mathbf{e}_1; \cdots; \mathbf{e}_K] \in \mathbb{R}^{K \times D}$, where K is the codebook size and D is the latent embedding dimension. Specifically, the idea is to associate each $\mathbf{h}_t^e \in \mathbf{H}_x$ with one $\mathbf{e}_{k(t)} \in \mathbf{H}_x$, where $k(t)$ is an index mapping function that maps the index t to another index $j \in [1, \cdots, K]$ (will be defined later). After this, the latent code for input $\mathbf{x}$ is then defined via an aggregation function (we define it as a "mean function" for simplicity):

$$\mathbf{z}_x \triangleq \frac{1}{T} \sum_{t=1}^{T} \mathbf{e}_{k(t)} \ . \tag{6}$$

There are several ways to define the index mapping $k(\cdot)$. We adopt the idea of nearest neighbor to define $k(\cdot)$ by choosing a code from $\mathbf{E}$ that is closest to $\mathbf{h}_t^e$ after a linear transformation. Formally, let $\tilde{\mathbf{h}}_t^e \triangleq \mathbf{W}_e \mathbf{h}_t^e + \mathbf{b}_e$ with learnable parameters $(\mathbf{W}_e, \mathbf{b}_e)$, we define $k(\cdot)$ as

$$k(t) \triangleq \arg\min_j \|\tilde{\mathbf{h}}_t^e - \mathbf{e}_j\|_2 \tag{7}$$

Based on the above construction, it is easy to see that given the codebook $\mathbf{H}$, the final latent code $\mathbf{z}_x$ of the input $\mathbf{x}$ can be formulated as a discrete distribution, *i.e.*,

$$q(\mathbf{z}_x|\mathbf{x}, \mathbf{H}) = \sum_{k=1}^{K} \alpha_k(\mathbf{x}) \delta_{\mathbf{e}_k}(\cdot) \ , \tag{8}$$

where $\alpha_k(\mathbf{x}) = \sum_{t=1}^{T} \mathbf{1}(\tilde{\mathbf{h}}_t^e = \mathbf{e}_k)/K$; and $\delta_{\mathbf{e}}(\cdot)$ is a delta function with point mass at $\mathbf{e}$. With such a construction, one can easily check that the KL-divergence between $q(\mathbf{z}_x|\mathbf{x}, \mathbf{H})$ and a prior distribution $p(\mathbf{z}_x)$ from the generative model (usually set to be from a simple Gaussian or uniform distribution) can be calculated as

$$\begin{aligned} &\mathrm{KL}(q(\mathbf{z}_x|\mathbf{x}, \mathbf{H})||p(\mathbf{z}_x)) \\ &= \sum_{k:\alpha_k>0} \alpha_k(\mathbf{x}) \left[\log(\alpha_k(\mathbf{x})) - \log(p(\mathbf{z}))\right] \ . \end{aligned} \tag{9}$$

Remark 1 *We observe that by optimizing the global codebook in the training process to make most of the codes informative so that they lie within the low-density regions of the prior $p(\mathbf{z})$, the KL-divergence will always be larger than zero, effectively preventing posterior collapse.*

Decoder Similar to the encoder, we parameterize the decoder with another LSTM. The target hidden state $\mathbf{h}_t^d$ can be progressively calculated as

$$\mathbf{h}_t^d = \mathbf{LSTM}([\mathbf{w}_{x_t}, \mathbf{z}_x], \mathbf{h}_{t-1}^d) \ , \tag{10}$$

Finally, we calculate the output distribution over the entire vocabulary at time t as

$$P_t = \mathrm{Softmax}(\mathbf{W}_o \mathbf{h}_t^d) \tag{11}$$

3.2 Training

Learning DB-VAE is divided into two parts: 1) learning the encoder and decoder; and 2) learning the global codebook.

Learning the encoder and decoder Our proposed DB-VAE model belongs to the general VAE framework, by defining a special form of the variational distribution as in (8). As a result, the encoder and decoder can be learned by optimizing the evidence lower bound (ELBO):

$$\begin{aligned} \mathrm{ELBO} = E_{q(\mathbf{x})}[&E_{q_\phi(\mathbf{z}_x|\mathbf{x},\mathbf{H})} \log p_\theta(\mathbf{x}|\mathbf{z}) \\ &- \mathrm{KL}(q_\phi(\mathbf{z}_x|\mathbf{x}, \mathbf{H})||p(\mathbf{z}))] \ , \end{aligned} \tag{12}$$

where $q(\mathbf{x})$ denotes the training data distribution; and the KL term is evaluated following (9).

Learning the codebook Directly optimizing the codebook with the above ELBO is infeasible because gradients cannot propagate back to the codebook due to the non-differentiable operator defined in (7). To this end, we follow (van den Oord et al., 2017) and define a new objective for updating the codebook. The key observation is that the codebook only appears in (7), thus the goal is to update the codebook such that it makes the distance between a latent code and the corresponding codebook atom minimal. Specifically, the loss is defined as

$$\begin{aligned} \mathcal{L}_{code} = \frac{1}{T} \sum_{t=1}^{T} (&\|\mathrm{sg}(\tilde{\mathbf{h}}_t^e) - \mathbf{e}_{k(t)}\|_2^2 \\ &+ \beta \|\tilde{\mathbf{h}}_t^e - \mathrm{sg}(\mathbf{e}_{k(t)})\|_2^2) \ , \end{aligned} \tag{13}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator to avoid complicated gradient flows and stabilize the training; β is a constant to balance the two terms.

The overall algorithm The full training algorithm is summarized in Algorithm 1. We find that it is important to balance between learning the encoder-decoder and learning the codebook. At the beginning, if the codebook does not learn as fast as the encoder, there will be a low utilization rate of the codebook to prevent codebook learning, *e.g.*, most of the input samples only focus on a limited atoms of the codebook. To overcome this issue, we add a strike-through pretraining step, where the decoder is fed with the latent codes directly from the encoder. This ensures that reasonable gradients can be passed through the latent space and the encoder. In the following, we will apply the superscript "(i)" on a variable (or function) to denote the dependency of the variable to the i-th input sample, *e.g.*, $\tilde{\mathbf{h}}_T^{(i)}$. To determine whether one should perform a pretraining step, we define a perplexity score ppl_code to monitor the utilization of the codebook:

$$\begin{aligned} \mathbf{v} &= \frac{1}{m}\sum_{i=1}^{m} \text{one\_hot}(k^{(i)}(t)) \\ ppl\_code &= \exp[-|\mathbf{v} \odot \log(\mathbf{v})|_1] \end{aligned} \tag{14}$$

where one_hot$(k(t))$ denotes a all-zero $1 \times K$ vector except the $k(t)$-th bit, which is set to 1. It is clear that the ppl_code value is large when the elements in $\mathbf{v}$ are close to uniform. Thus it can be used to indicate the utilized rate of the codebook.

Algorithm 1: DB-VAE training

Require: encoder f_ϕ, decoder g_θ, codebook $\mathbf{E}$, threshold σ and batch size m
Step 1: Strike-through pretraining
while $ppl\_code \leq \sigma$ **do**
  Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim q(\mathbf{x})$
  Compute $\tilde{\mathbf{h}}_T^{(i)} = f_\phi(\mathbf{x}^{(i)}), \tilde{\mathbf{x}}^{(i)} = g_\theta(\tilde{\mathbf{h}}_T^{(i)})$
  Optimize the ELBO(θ, ϕ) to train f_ϕ, g_θ
  Optimize $\mathcal{L}_{code}$ to learn $\mathbf{E}$
Step 2: Joint training
while *done* **do**
  Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim q(\mathbf{x})$
  Compute $\mathbf{z}_x^{(i)}, \tilde{\mathbf{x}}^{(i)} = g_\theta(\mathbf{z}_x^{(i)})$
  Optimize $\mathcal{L}_{code}$ to learn $\mathbf{E}$
  Backprop -ELBO+$\mathcal{L}_{code}$ to train f_ϕ, g_θ

Extension: *top-k* NN search In our construction of a latent code, we search the nearest code from the codebook via the index mapping defined in (7). Such a construction endows a limitation where a hidden state from the LSTM only corresponds to one atom from the codebook. This scheme, however, does not fit real applications well. For example, in neural machine translation, one source sentence (one hidden state) can correspond to multiple correct translations (multiple atoms); and in dialog response generation, a good model should be able to generate multiple relevant and diverse responses when same contexts are given. Furthermore, when a VAE is well trained, input texts with similar semantics should be mapped to close clusters in the latent space (see Section 5.1). As a result, we propose a generalization by extending the 1-NN search to k-NN search when searching the codebook to construct latent codes. In other words, Eq. (7) returns a set of k indexes, corresponding to the k nearest codebook atoms from the codebook. These atoms are then averaged over the whole sequence to generate the final latent code, as in Eq.(6). The corresponding algorithm is summarized in Algorithm 2.

Algorithm 2: *top-k* NN Search Extension

Result: $\{\tilde{\mathbf{x}}^{(i)}\}_{i=1}^k$
Require:encoder f_θ, decoder g_ϕ, codebook $\mathbf{E}$
while *done* **do**
  Sample $\mathbf{x} \sim q(\mathbf{x})$
  Find k-NN instead of 1-NN as in Eq.(7) to calculate $\{\mathbf{z}_x^{(i)}\}_{i=1}^k$
  Generate $\{\tilde{\mathbf{x}}^{(i)} = g_\theta(\mathbf{z}_x^{(i)})\}_{i=1}^k$

4 Related Work on Posterior Collapse

Several attempts have been made to alleviate the posterior-collapse issue. Among them, perhaps the simplest solution is via KL cost annealing, where the weight of the KL penalty term is scheduled to gradually increase during training (Bowman et al., 2015). Later, Fu et al. (2019) proposes a cyclical annealing schedule, which allows progressive learning of more meaningful latent codes by leveraging informative representations of previous cycles as warm re-starts. These approaches tend to manually encourage the use of latent codes, but might hurt a model's density approximation ability as pointed out in (He et al., 2019). Our method differs from these methods in that it maintains a model's representation power while learning an informative latent space.

Other solutions include weakening the capacity of a generative network or enhancing the inference network. Yang et al. (2017) proposes the use of a dilated CNN as a decoder in VAE by controlling the size of context from previously generated

words. (Kim et al., 2018) propose a semi-amortized approach that uses stochastic variation inference to iteratively refine an inference network. This method, however, is expensive to train. Similarly, He et al. (2019) propose a simple yet effective training algorithm that aggressively optimizes the inference network with more updates. Other threads of solutions introduce more complicated priors in the latent space (Tomczak and Welling, 2017; Xu and Durrett, 2018). Makhzani et al. (2015); Joulin et al. (2016) further replace the KL regularizer with an adversarial regularizer. Our work outperforms these methods without increasing additional training burdens.

In the case of discrete representations in VAE, the most related work is (Zhao et al., 2018). It applies the Gumbel-Softmax trick (Jang et al., 2016) to train discrete variables, resulting in effective and interpretable dialog generation. Our approach has wider applicability and is ready to be extended to more NLP tasks. Other approaches combine vector quantization and the Transformer model (Kaiser et al., 2018; Roy et al., 2018). These approaches have primarily focused on non-autoregressive neural machine translation, which did not investigate the posterior collapse issue in sequential variational inference.

5 Experiments

We conduct extensive experiments to demonstrate the effectiveness and efficiency of the proposed DB-VAE on various language processing tasks, including language modeling (LM), unaligned text-style transfer, dialog-response generation and neural machine translation (NMT). In addition, we also evaluate how the codebook size K will affect a model's performance. Code for reproducing these results will be made publicly available.

5.1 Language modeling

Following (Yang et al., 2017), we evaluate our model for language modeling mainly on two large-scale document corpus, *Yahoo* and *Yelp*. Detailed statistics of the two datasets are given in Table 7 in the Supplementary Material (SM) A.1. We first used a simple *synthetic* dataset (He et al., 2019) consisting of 16k training sentences and 4k testing sentences to evaluate how the codebook size affects the model's performance.

The impact of codebook size K We first investigate the impact of codebook size K on the model's behavior. The learning curves with different K are shown in Figure 7 of the SM A.1. Because validation *ppl*'s are very close when $K \geq 2^{16}$, we adopt $K = 2^{16}$ in all our experiments (a trade-off between memory and performance) unless explicitly declared.

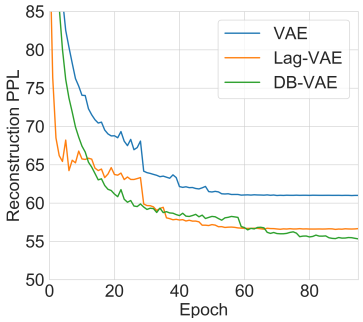

Figure 2: Learning curves of VAE, Lag-VAE and DB-VAE on *Yahoo*.

Baseline and training details Four representative LM models are chosen as baselines, including LSTM-LM, the standard VAE, SA-VAE (Kim et al., 2018) and Lag-VAE (He et al., 2019), the current state-of-the-art. For fair comparisons, both the recognition network and generative network are implemented as a 1-layer LSTM with 1024 hidden units for all models. The word embedding dimension is set to 1024 and the latent dimension to 32. The SGD optimizer with the same setting is applied to all models. The latent variable is used to initialize the hidden state of the decoder and fed as additional input at each time step.

LM results The results in terms of reconstruction error, perplexity and training time are shown in Table 1 and in Figure 2. As expected, our model achieves the best performance in all the metrics due to the flexibility of the discrete variational distribution, which makes the model free of posterior collapse. Remarkably, our model runs almost as fast as the standard VAE. The faster convergence of Lag-VAE at the beginning is because it aggressively trains an encoder, where approximately $50\times$ more data are used to train the encoder in one epoch.

Latent space visualization For better understanding, we visualize the latent representations of the whole dataset using t-SNE projection (Maaten and Hinton, 2008) in Figure 3. It is seen that our model is able to learn a much smoother and more separable transition from 0-star to 4-star reviews. To visualize the codebook utilization, we also compute the $\mathbf{v}$ (Eq.14) on a random batch of testing

Table 1: Performance comparisons on language modeling on the *Yelp and Yahoo* corpus.

Models	Rec(KL)	Rec-PPL	Time
Yelp corpus			
LSTM-LM	358.1	40.64	-
VAE	357.9	40.56	5.4
SA-VAE	357.5	40.39	56.3
Lag-VAE	351.4	37.92	20.3
DB-VAE	**349.7**	**37.26**	5.4
Yahoo corpus			
LSTM-LM	328.0	60.75	-
VAE	328.6	61.21	6.9
SA-VAE	329.1	61.59	69.2
Lag-VAE	322.6	56.78	15.30
DB-VAE	**320.4**	**55.24**	7.0

data after each training epoch. As shown in Figure 4, the usage of the codebook becomes more balanced as the training goes on.

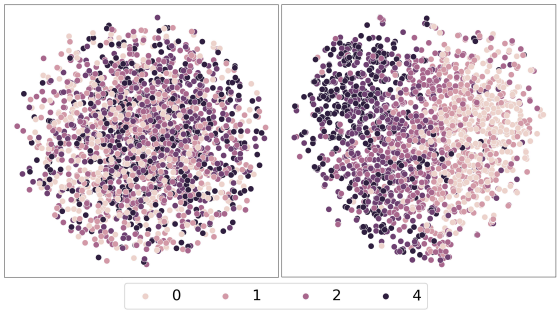

Figure 3: t-SNE embeddings of latent space on *Yelp* corpus. Left: Lag-VAE, Right: DB-VAE. 0-4 represents the review score, from negative to positive.

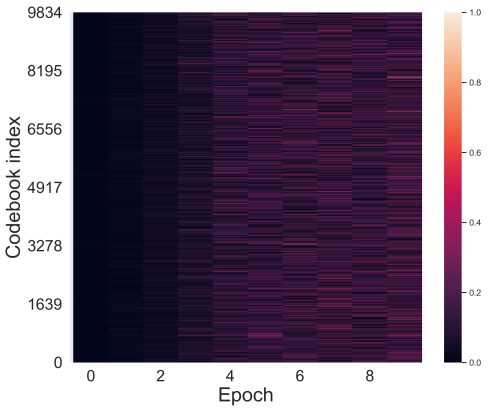

Figure 4: The heatmap of codebook learning on *Yelp*. The x-axis corresponds to the training epoch, and y-axis corresponds to indices of different codes.

Codebook interpolation Particularly in text modeling, when performing a convex combination between any two latent codes $\mathbf{z}_1$ and $\mathbf{z}_2$, the interpolation is equivalent to $\tilde{\mathbf{x}}_\lambda = g_\phi(\lambda \mathbf{z}_1 + (1-\lambda)\mathbf{z}_2)$. Ideally, adjusting λ from 0 to 1 will generate a series of sentences, where $\mathbf{x}_\lambda$ will be less semantically similar with the sentence corresponding to $\mathbf{z}_1$ and much more semantically similar to that of $\mathbf{z}_2$ (Berthelot et al., 2018). Table 2 shows the generated sentences when λ ranges from 0.0 to 1.0 with a stepsize of 0.2. Indeed, intermediate sentences $\mathbf{x}_\lambda$ produced by the proposed model can provide a semantically smooth morphing between the two endpoints. More detailed examples are provided in Table 8 in the SM A.

5.2 Unaligned neural text style transfer

Next, we evaluate the proposed model on the unaligned sentiment transfer task on the *Yelp* dataset. Review ratings above three are considered positive, and those below three are considered negative. Hence, we split the corpus into two sets of unaligned positive reviews (350k) and negative reviews (250k). The goal of the style transfer task is to change the underlying sentiment between positive and negative reviews.

Experiment setup We denote y as the sentiment attribute and construct a decoder to implement the conditional distribution $p(\mathbf{x}|\mathbf{z}, y)$. Following the setup in (Zhao et al., 2017a; Shen et al., 2017), we train two separate decoders where one is for positive reviews, $p(\mathbf{x}|\mathbf{z}, y = 1)$, and the other one is for negative reviews, $p(\mathbf{x}|\mathbf{z}, y = 0)$. Normally, the latent prior $p(\mathbf{z})$ will encode all the semantic and attribute information of the input. In the models, we want the attribute information to be excluded from $p(\mathbf{z})$ and let the decoder learns to produce the transferred reviews. According to (Zhao et al., 2017a), a classfier c_ψ is introduced to distinguish the latent code's attribute, and adversarially train the encoder to fool the classifier and thus remove the sentiment attribute from the latent space.

Baseline We compare our model with two strong baselines: 1) an adversarially regularized autoencoder (ARAE) (Zhao et al., 2017a), which learns the prior $p(\mathbf{z})$ via a more expensive and unstable adversarial training; 2) a recently developed implicit deep-latent-variable model (iVAE) (Fang et al., 2019) that applies sample-based representations of variational distributions.

Quantitative metrics We adopt several quantitative metrics: (i) Transfer: it measures the style transfer accuracy evaluated on an automatic classification model (*fastText* library (Joulin et al., 2016)); (ii) BLEU: the consistency between the translated

Table 2: Interpolating between latent codes

λ	**Generated intermediate sentences**
0.0	i had a great experience with the staff and the staff was very friendly and helpful ! i will definitely...
0.2	stopped in for a quick bite before heading out to the airport . i would definitely recommend...
0.4	stopped in for a quick bite before heading out to the airport . the service was fast and friendly...
0.6	my husband and i stopped in for a quick bite before heading out to the airport . we were seated...
0.8	this was my first time here and i will definitely be back . the service was good , the food was good...
1.0	this place was pretty good . i had the pulled pork sandwich and it was pretty good , but nothing...

Table 3: Performances on *Yelp* sentiment transfer

Model	Transfer↑	BLEU↑	PPL↓	RPPL↓
ARAE	95.0	32.5	6.8	395
iVAE	92.0	36.7	6.2	285
DB-VAE	**97.1**	**40.2**	**4.8**	**254**

Table 4: Sentiment transfer results on *Yelp*

Negative ⇒ Positive	
Input	the staff was very rude as well .
DB-VAE	the staff here is also fantastic .
ARAE	the staff was very friendly .
Input	but , the food is not good .
DB-VAE	but , the food and brews are the best .
ARAE	well, nice atmosphere with a nice selection .
Input	just had a bad experience with a _num_ minutes.
DB-VAE	always a great spot for happy hour or lunch .
ARAE	i love their happy hour .
Positive ⇒ Negative	
Input	but , it 's worth it !
DB-VAE	however , it 's just ok .
ARAE	but , i was so disappointed .
Input	the food is always fresh and tasty .
DB-VAE	the food was n't good , and not fresh .
ARAE	the food was not good but the food was not very good .
Input	the service was top notch and so was the food .
DB-VAE	the service was slow and the food was very slow.
ARAE	i was told the server was nice but the food was cold .

candidate and the original reference; (iii) PPL and Reverse PPL (RPPL): PPL measures the fluency of the generated text, and RPPL works in a reverse fashion, which is computed by training an LM on generated data and evaluated on the original data. Mode collapse may be detected by the RPPL value.

Quantitative analysis Table 3 shows the sentiment transfer results. The proposed method outperforms ARAE in all metrics. On the one hand, in addition to the higher PPL and RPPL, our model preserves the superiority that has already been highlighted in Section 5.1. On the other hand, compared with ARAE, the higher transfer accuracy, and BLEU score indicate that our model can capture more sentiment related information while keeping the grammar structure in the original text and the opposite text consistent.

Qualitative results Some randomly selected examples are give in Table 4. It can be observed that both ARAE and DB-VAE can successfully transfer the sentiment given the input. However, DB-VAE shows better capability in content preserving, and this observation is per the BLEU scores in Table 3.

5.3 Dialog response generation

In this experiment, we follow (Gu et al., 2018) and evaluate the proposed model on two widely-used dialog datasets *Switchboard* (Godfrey and Holliman, 1997) and *DailyDialog Dataset* (Li et al., 2017). Responses generated by VAE-based models (Zhao et al., 2017c; Gu et al., 2018) are conditioned on the latent variable. So, this task can examine whether a model can capture a richer latent space and thus generate more diverse, informative and consistent responses.

Baselines We compare our model's performance with five representative baselines for dialog modeling: (i) SeqGAN: a GAN-based model for sequence generation (Yu et al., 2017); (ii) CVAE: a conditional VAE model (Zhao et al., 2017c); (iii) CVAE-BOW: CVAE with bag-of-word loss (Zhao et al., 2017c); (iv) VHRED: a hierarchical VAE model (Serban et al., 2017); (v) WAE-GMP: a conditional Wasserstein autoencoder with a Gaussian mixture prior network (Gu et al., 2018), which holds the state-of-the-art. (vi) DI-VAE: a discrete VAE which is most related to our work.

Quantitative metrics Follow the evaluation setup in (Gu et al., 2018), three evaluation metrics (see details in A.2) are used:
(i) Sentence-level BLEU, which works by counting n-grams in the candidate (generated) sentences to n-grams in the reference text. (ii) BOW Embedding, which calculates the cosine similarity of bag-of-words embedding between the candidate and the reference. (iii) Distinct, which computes the diversity of the generated responses.

Table 5: Performance comparison on dialog response generation, *Switchboard Dataset*

Model	BLEU↑			BOW Embedding↑			intra-dist↑		inter-dist↑	
	R	P	F1	A	E	G	dist-1	dist-2	dist-1	dist-2
SeqGAN	0.282	0.282	0.282	0.817	0.515	0.748	0.705	0.521	0.070	0.052
CVAE	0.295	0.258	0.275	0.836	0.572	0.846	0.803	0.415	0.112	0.102
CVAE-BOW	0.298	0.272	0.284	0.828	0.555	0.840	0.819	0.493	0.107	0.099
VHRED	0.253	0.231	0.242	0.810	0.531	0.844	0.881	0.522	0.110	0.092
WAE-GMP	0.420	0.258	0.319	0.925	0.661	0.894	0.713	0.671	0.333	0.555
DI-VAE	0.310	0.175	0.224	0.802	0.583	0.862	0.891	0.779	0.489	0.767
DB-VAE	0.386	**0.274**	**0.320**	**0.925**	**0.668**	**0.906**	**0.905**	**0.836**	**0.553**	**0.808**

Quantitative analysis Table 5 and Table 9 show the quantitative results of our model and other strong baselines on *Switchboard* and *DailyDialog*. Our model outperforms the baselines in most metrics. Although our method obtains a similar BLEU score as WAE-GMP, the inter-dist and intra-dist scores are much higher. In terms of intra-dist, the dist-1 and dist-2 on *Switchboard* are 19.2% and 24.6% higher than WAE-GMP. This indicates that our model is capable of generating less repeated n-grams in each response. As for the inter-dist, dist-1 and dist-2 are even 66.1% and 45.6% higher than WAE-GMP, meaning that our model generates much more diverse responses than WAE-GMP.

Table 6: Evaluation on NMT, *IWLST14*

Model	PPL	BLEU
Variational Attention	6.13	33.41
RNNsearch	5.72	33.29
RNNsearch w/ top-k (ours)	**5.63**	**33.59**

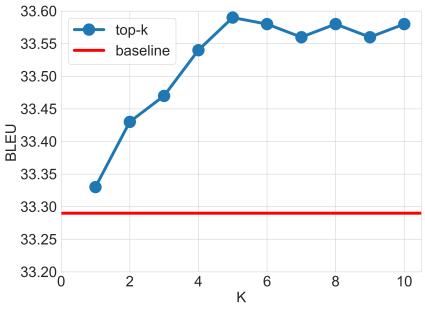

Figure 5: BLEU score on IWLST14

5.4 Extension: RNN-based NMT model

We finally evaluate our model with the proposed *top-k* NN search on the German-English translation task. Our model is built on a baseline RNNsearch architecture (Bahdanau et al., 2014). The recently proposed variational attention model (Deng et al., 2018) is also adopted as a baseline.

We use the IWLST14 dataset (Cettolo et al., 2014), which is a standard benchmark for experimental NMT models. This dataset contains around 153K, 7K and 7K sentences for training, validation and testing, respectively. The same preprocessing as in (Ott et al., 2018) is applied. As for the architecture, both the encoder and the decoder have one layer, each with 512-dimensional embedding. For BLEU evaluation, the beam size in beam search is 5. The library *Fairseq* (Ott et al., 2019) is adopted as the codebase. The codebook size K is set to 2^{20}, and only the final hidden state of the encoder passes through the discretized bottleneck.

Results averaged by 5 different runs are reported in Table 6 and in Figure 5. Note the attention mechanism is used in RNNsearch, where each progressed state in the decoder side has direct access to the state in the encoder side. Although we only discretize the final hidden state of the encoder as formulated in Section 3.1, a notable improvement on the PPL and BLEU score is still observed. Following Algorithm 2, as we increase the value of k from 1 to 10, the BLEU score continues increasing until k reaches 5. The reason might be that the top-5 latent codes have already encoded most source-target combinations. Besides, the BLEU score is as low as 26.1 when we choose the farthest latent code from the codebook instead. These validate the effectiveness of our proposed top-k inference strategy which applies to most RNN-based autoencoder models.

6 Conclusion

We propose the DB-VAE, a variant of VAE that uses a discretized bottleneck obtained from a global codebook for latent representations. Our model can potentially overcome the posterior collapse issues in Seq2Seq models. The proposed DB-VAE can provide a good balance between optimization of the inference network and the generative network. Moreover, our DB-VAE can also interpret richer semantic information of discrete structured sequences. Extensive experiments demonstrate the effectiveness of the proposed approach. DB-VAE is flexible enough to be extended to other NLP models such as the Transformer and BERT, which are left as interesting future work.

References

Alexander A Alemi, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy. 2017. Fixing a broken elbo. *arXiv preprint arXiv:1711.00464*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. 2018. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*.

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

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, page 57.

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9712–9724.

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

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*.

Otto Fabius and Joost R van Amersfoort. 2014. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. *arXiv preprint arXiv:1908.11527*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, Lawrence Carin, et al. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*.

John J Godfrey and Edward Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*, 926:927.

Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. 2018. Dialogwae: Multimodal response generation with conditional wasserstein autoencoder. *arXiv preprint arXiv:1805.12352*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Matthew D Hoffman and Matthew J Johnson. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

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

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

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pages 236–244.

Oluwatobi Olabiyi and Erik T Mueller. 2019. Multi-turn dialogue response generation with autoregressive transformer models. *arXiv preprint arXiv:1908.01841*.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. 2018. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Harshil Shah and David Barber. 2018. Generative neural machine translation. In *Advances in Neural Information Processing Systems*, pages 1346–1355.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. 2019. Variational autoencoder with implicit optimal priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5066–5073.

Jakub M Tomczak and Max Welling. 2017. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org.

Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. 2017. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2017a. Adversarially regularized autoencoders. *arXiv preprint arXiv:1706.04223*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017b. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*.

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

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017c. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

A Supplemental Material

A.1 Language modeling

Table 7: Statistics of LM datasets

Corpus	#vocabulary	#sentences	avg.length
Yahoo	20001	10000	80
Yelp	19997	10000	97

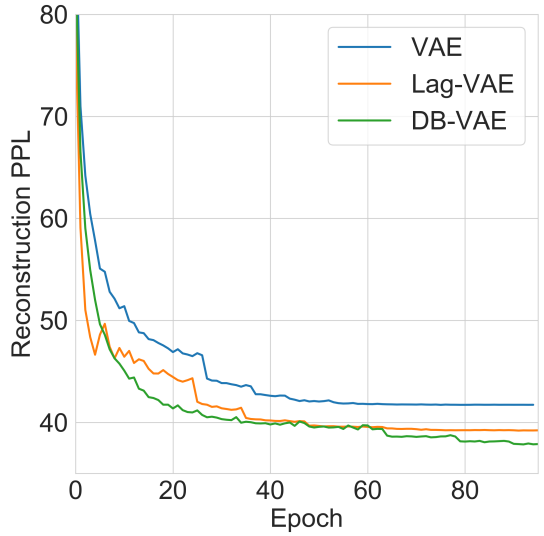

Figure 6: Learning curves of VAE, Lag-VAE and DB-VAE on *Yelp*

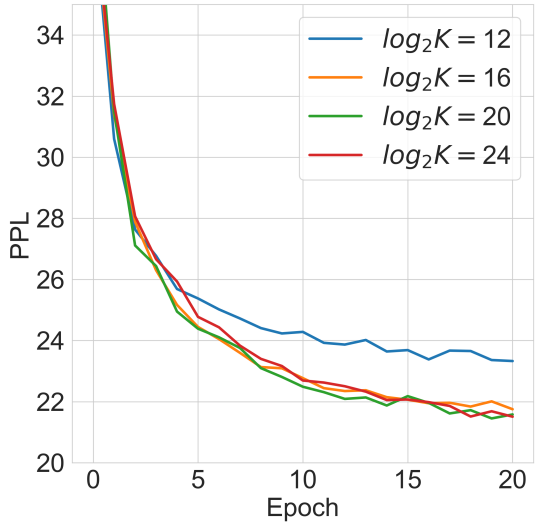

Figure 7: Learning curves with differnt codebook size K

A.2 Dialog response generation

Detailed evaluation metrics used in dialog-response-generation task:
(i) Sentence-level BLEU, which works by counting n-grams in the candidate (generated) sentences to n-grams in the reference text. To compute the score, the setting is identical WAE-GMP (Gu et al., 2018) where 10 responses (candidates) are sampled from the models for each test context. K is set to 10 in Algorithm 2. The precision and recall of BLEU are defined in (Zhao et al., 2017c).
(ii) BOW Embedding, which calculates the cosine similarity of bag-of-words embedding between the candidate and the reference. We adopt three metrics here to compute the similarity, *greedy* (Rus and Lintean, 2012), *average* (Mitchell and Lapata, 2008) and *extreme* (Forgues et al., 2014).
(iii) Distinct, which computes the diversity of the generated responses. dist-n is defined as the ratio of unique n-grams (n=1,2) over all n-grams in the generated responses. As multiple responses are sampled from the models, we can define intra-dist as the average of distinct values within each sampled response and inter-dist as the distinct value among all sampled responses.

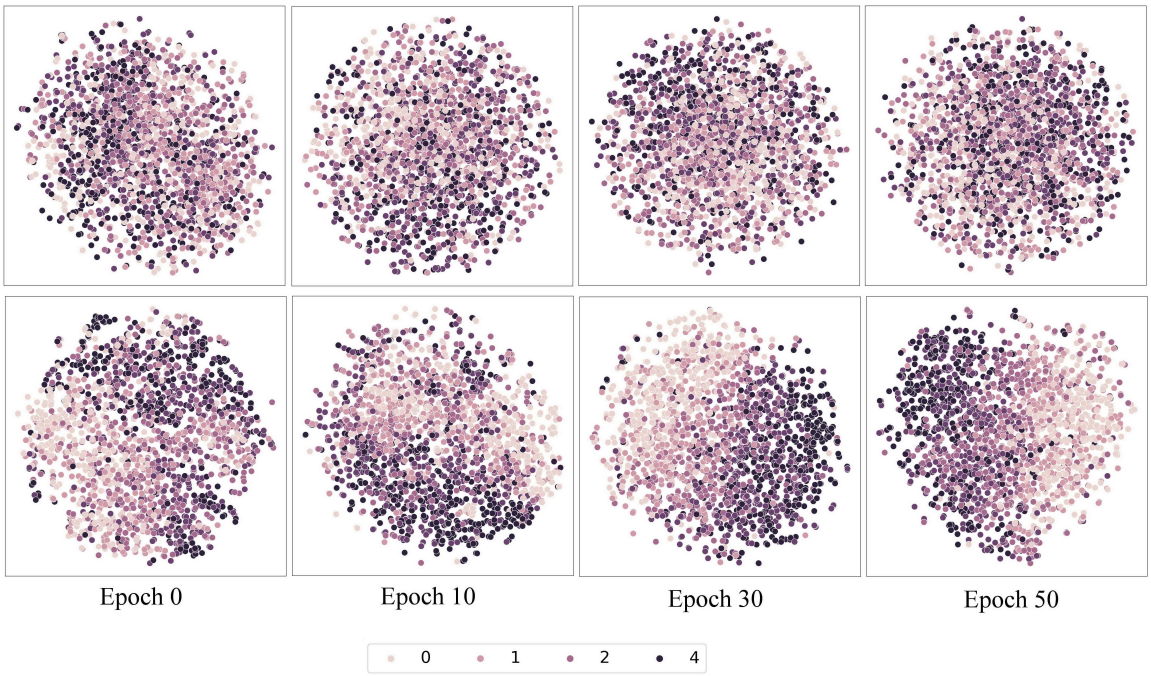

Figure 8: t-SNE embeddings of latent space on *Yelp* corpus. Top: Lag-VAE, Bottom: DB-VAE. 0-4 represents the review score, from negative to positive.

Table 8: Detailed interpolation results

λ	**Generated intermediate sentences**
0.0	had a great experience at this place ! i had a great experience with the staff and the staff was very friendly and helpful ! i had a great experience and i will definitely be back !
0.1	had a great experience here ! the staff was very friendly and helpful ! i had a great time and i will definitely be back !
0.2	stopped in for a quick bite before heading out to the airport . i had the chicken and waffles and it was delicious ! i would definitely recommend this place to anyone looking for a great breakfast !
0.3	stopped in for a quick bite before heading out to the airport . i had the chicken and waffles and it was delicious ! the service was fast and friendly . i will definitely be back !
0.4	stopped in for a quick bite before heading out to the airport . i had the chicken and waffles and it was delicious ! the service was friendly and fast . i 'll be back !
0.5	my husband and i stopped in for a quick bite before heading out to the airport . we were seated right away and we were seated right away . our server was very friendly and helpful . the food was pretty good and the service was great .
0.6	my husband and i stopped in for a quick bite before heading out to the airport . we were seated right
0.7	this was my first time here and i will definitely be back . the service was fast and friendly and the food was delicious . i 'll be back .
0.8	this was my first time here and i will definitely be back . the service was good , the food was good , and the prices were reasonable . i 'll be back .
0.9	this place was pretty good . i had the chicken and waffles and it was pretty good . i 'd definitely go back .
1.0	this place was pretty good . i had the pulled pork sandwich and it was pretty good , but nothing special . the fries were pretty good though .

Table 9: Performance comparison on dialog response generation, *DailyDialog Dataset*

Model	BLEU↑			BOW Embedding↑			intra-dist↑		inter-dist↑	
	R	P	F1	A	E	G	dist-1	dist-2	dist-1	dist-2
SeqGAN	0.270	0.270	0.270	0.907	0.495	0.774	0.747	0.806	0.075	0.081
CVAE	0.265	0.222	0.242	0.923	0.543	0.811	0.938	0.973	0.177	0.222
CVAE-BOW	0.256	0.224	0.239	0.923	0.540	0.812	0.949	0.976	0.165	0.206
VHRED	0.271	0.260	0.265	0.892	0.507	0.786	0.633	0.711	0.071	0.089
WAE-GMP	0.372	0.286	0.323	0.952	0.591	0.853	0.754	0.892	0.313	0.597
DI-VAE	0.323	0.190	0.239	0.874	0.600	0.814	0.947	0.963	0.500	0.718
DB-VAE	**0.373**	0.276	0.317	0.944	**0.615**	0.839	**0.954**	**0.997**	0.467	**0.787**