Scale-VAE: Preventing Posterior Collapse in Variational Autoencoder

Tianbao Song[1], **Jingbo Sun**[2*], **Xin Liu**[3], **Weiming Peng**[2]
[1] School of Computer and Artificial Intelligence, Beijing Technology and Business University, China
[2] School of Artificial Intelligence, Beijing Normal University, China
[3] The 15th Research Institute of China Electronics Technology Group Corporation, China
songtianbao@btbu.edu.cn, sunjingbo@mail.bnu.edu.cn, threetwoli@163.com, pengweiming@bnu.edu.cn

Abstract

Variational autoencoder (VAE) is a widely used generative model that gains great popularity for its capability in density estimation and representation learning. However, when employing a strong autoregressive generation network, VAE tends to converge to a degenerate local optimum known as posterior collapse. In this paper, we propose a model named Scale-VAE to solve this problem. Scale-VAE does not force the KL term to be larger than a positive constant, but aims to make the latent variables easier to be exploited by the generation network. Specifically, each dimension of the mean for the approximate posterior distribution is multiplied by a factor to keep that dimension discriminative across data instances. The same factors are used for all data instances so as not to change the relative relationship between the posterior distributions. Latent variables from the scaled-up posteriors are fed into the generation network, but the original posteriors are still used to calculate the KL term. In this way, Scale-VAE can solve the posterior collapse problem with a training cost similar to or even lower than the basic VAE. Experimental results show that Scale-VAE outperforms state-of-the-art models in density estimation, representation learning, and consistency of the latent space, and is competitive with other models in generation.

Keywords: variational autoencoder, posterior collapse, representation learning, latent space

1. Introduction

As a widely used generative model, variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) gains great popularity for its capability in density estimation and representation learning. VAE defines the joint distribution between the observed data and a set of latent variables, and approximates the posterior of latent variables using the amortized variational inference. Despite the successful use in various challenging domains, a notorious problem of VAE is that it often ignores the latent variables completely, particularly when employing the strong autoregressive generation networks, such as LSTM (Hochreiter and Schmidhuber, 2012) on text or PixelCNN (Van den Oord et al., 2016) on images. In this case, VAE fails to diversify the posteriors of different data by simply using the single posterior distribution that is almost identical to the prior to model all data instances, which is termed the posterior collapse or the KL (Kullback–Leibler divergence) vanishing problem (Bowman et al., 2016).

Many convincing solutions have been proposed to alleviate the problem of posterior collapse. One thread of research is to weaken the generation network by replacing it with a less autoregressive alternative (Semeniuta et al., 2017; Yang et al., 2017) or removing some of the conditional information (Bowman et al., 2016; Chen et al., 2016; Petit and Corro, 2021). However, these methods do not make full use of the expressive generation network, which may reduce the generative ability of the model. Another thread of research is to attribute the posterior collapse to optimization challenges and design various training strategies including KL annealing (Bowman et al., 2016; Fu et al., 2019), β-VAE (Higgins et al., 2016), Free-Bits (Kingma et al., 2016), semi-amortized VAE (Kim et al., 2018), aggressive training (He et al., 2018), and inference network pretraining (Li et al., 2019). However, these methods either do not address the essence of the problem, have non-smooth optimizations (Chen et al., 2016), or are very time consuming.

Some other studies attempt to boost the mutual information (MI) between latent variables and input data by adding additional terms to the objective (Zhao et al., 2019; Zheng et al., 2019). However, due to the intractability of approximating the MI-involved objective, additional optimization methods such as maximum-mean discrepancy or Gibbs inequality are required. Ma et al. (2019) introduced mutual posterior-divergence regularization in the objective, which is analytical and has a similar goal with MI. However, the relative scale of the regularization term and original objective requires deliberate tuning.

Another important thread is to keep the KL term of the objective as a positive constant. Some studies

*Corresponding author

use other distributions instead of the Gaussian prior, such as von Mises-Fisher distribution (Davidson et al., 2018; Guu et al., 2018) or uniform distribution (Van den Oord et al., 2017; Zhao et al., 2018). δ-VAE (Razavi et al., 2019) constrains the mean and variance of the posterior to have a minimum distance to the prior. However, forcing the same constant KL for all data instances or setting the parameters in a specific range may limit the model performance. A batch normalization-based method has achieved encouraging results (Zhu et al., 2020; Shen et al., 2021). It can avoid the posterior collapse problem by keeping the expectation of the KL's distribution positive. However, the theoretical basis was not clearly stated, and keeping the KL term positive is not a sufficient condition to avoid posterior collapse. In addition, this method may lead to semantic confusion in the latent space.

In this paper, we present a novel model named Scale-VAE that can avoid the posterior collapse effectively and efficiently. The contributions can be summarized as follows:

- We analyze the causes of posterior collapse in VAE and accordingly summarize several directions to solve the problem.

- We propose a model named Scale-VAE in which a factor is multiplied to each dimension of the mean for the posterior distribution to keep the variance of the mean in that dimension not close to zero. The scaled-up latent variables are fed into the generation network, but the original posteriors are still used to calculate the KL term. In this way, the posterior collapse problem is solved and the generative ability of the model is guaranteed. Meanwhile, the same factors are used for all the data instances, which ensures the semantic consistency and coherence of the latent space.

- We make connections between Scale-VAE and previous work. Scale-VAE achieves the same goal as MI-based methods but is more effective and efficient. It tries to increase the divergence among the posterior distribution family and tries to maximize the MI between latent variables and input data.

- Extensive experiments have been conducted, and the results clearly show that Scale-VAE can outperform or be on par with state-of-the-art models on density estimation, representation learning, and latent space property.

2. Background and Related Work

VAE (Kingma and Welling, 2014; Rezende et al., 2014) aims to construct a smooth latent space $z \in Z$ by learning a generative model $p_\theta(x, z)$ to maximize the marginal likelihood $\log p_\theta(x) = \log \int_Z p_\theta(x|z)p(z)dz$ on the observed data $p(X)$. Typically, the prior $p(z)$ is assumed as the multivariate Gaussian distribution $N(0, I)$. Due to the intractability of this marginal likelihood, an amortized inference distribution $q_\phi(z|x)$ is utilized to approximate the true posterior. Then it turns out to optimize the evidence lower bound (ELBO):

$$L_{ELBO} = E_{p(X)}[E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\phi(z|x)\|p(z)]], \tag{1}$$

where $q_\phi(z|x)$ is parameterized as a multivariate Gaussian distribution $N(\mu, \sigma^2)$ by an inference network (encoder) with parameters ϕ, and $p_\theta(x|z)$ denotes the generation network (decoder) with parameters θ.

In practice, when applying autoregressive models as the decoder, e.g., LSTM that is a common choice in modeling text, the posteriors $q_\phi(z|x)$ of different data instances $x \in X$ tend to collapse to the prior $p(z)$. In this case, the latent variables z are independent of the data x, then VAE fails to learn meaningful representations of the data and the latent variables give no guidance to generation process.

Now let's focus on the cause of the posterior collapse problem. From the perspective of information bottleneck theory (Alemi et al., 2016), the latent variable z fed into the decoder is obtained from the posterior $q_\phi(z|x) = N(\mu, \sigma^2)$ using the reparameterization trick: first sample a noise variable ε from the Gaussian distribution $N(0, I)$; then compute z through a linear transformation using μ and σ. Thus, the mean μ and diagonal covariance matrix σ^2 transmit the information about the input data x. According to Eq. 1, VAE is trained by jointly maximizing the reconstruction term $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$ and minimizing the KL term $D_{KL}[q_\phi(z|x)\|p(z)]$. The KL term can be considered as an upper bound on the amount of the transmitted information. Then, there is a contradiction that the reconstruction term encourages the latent variables to transmit more information about the input data x, but the KL term limits the amount of information that can be transmitted.

In addition to this contradiction, during the early stages of training, the latent variables z contain little information about the input data x. This makes it difficult for VAE to exploit the information in z. When equipped with an expressive decoder, VAE tends to give up difficult latent variables and rely entirely on the decoder for generation, eventually converging to a degenerate local optimum.

According to the above analysis, solving the posterior collapse problem can be attempted from the following directions.

First, weaken the decoder, forcing the model to rely on latent variables for generation. Semeniuta et al. (2017) and Yang et al. (2017) implemented the decoder by CNN instead of autoregressive modeling. Bowman et al. (2016) randomly removed some fraction of the conditioned-on word tokens to weaken the decoder. Petit and Corro (2021) introduced a regularization term based on fraternal dropout in the objective that forces the hidden states computed by LSTM to be similar even if different word tokens in the input are masked. Chen et al. (2016) fed a lossy representation of data to the autoregressive decoder.

Second, alleviate the contradiction between the reconstruction term and the KL term, or use a looser constraint on the upper bound of the information transmitted by latent variables. Bowman et al. (2016) proposed the KL annealing method that gradually increases the weight of KL term during training. Fu et al. (2019) proposed to repeat the process of increasing the KL weight multiple times. Higgins et al. (2016) proposed β-VAE that re-weights the KL term using a hyperparameter β. Alemi et al. (2018) demonstrated that setting $\beta < 1$ avoids the posterior collapse problem, but setting $\beta \neq 1$ results in an improper statistical model. Kingma et al. (2016) proposed Free-Bits that replaces the KL term with a hinge loss term and stops optimization of the KL value when it is smaller than a threshold. Pelsmaeker and Aziz (2020) proposed the minimum desired rate technique to attain ELBO values at a pre-specified rate that does not suffer from the gradient discontinuities of Free-Bits. Davidson et al. (2018) and Guu et al. (2018) replaced the Gaussian prior with the von Mises-Fisher distribution so that the KL term is independent of the data instances. Song et al. (2022) specified the prior as multiple Gaussian distributions to alleviate the control over information upper bound transmitted in the latent variables. Havrylov and Titov (2020) proposed Levenstein VAE in which the generator disregarding latent variables will incur large penalties.

Third, add MI-based terms to the objective in order to enforce the relation between latent variables and input data. Zheng et al. (2019) proposed Fisher autoencoder that can implicitly control the MI between latent variables and input data by setting appropriate Fisher information constraint. Zhao et al. (2019) added a KL divergence term between the aggregated posterior and prior on latent variables to the objective and adjusted the scaling parameters of this term and the original KL term to control the MI. Ma et al. (2019) proposed to use a mutual posterior-divergence regularization in the objective, which has a similar goal with MI.

Fourth, reduce the difficulty of exploiting latent variables, especially in the initial stages of training. Dieng et al. (2019) proposed to add skip paths between the latent variables and the hidden layers of decoder. He et al. (2018) proposed to update the encoder with additional training loops before updating the decoder. Li et al. (2019) proposed to initialize the encoder of VAE with a pre-trained one from an autoencoder, and then continue training with Free-Bits. Kim et al. (2018) proposed to use amortized variational inference to initialize VAE and run stochastic variational inference to refine them. Zhu et al. (2020) proposed to apply batch normalization to the parameters of the approximate posteriors for latent variables and pointed out that keeping the expectation of KL positive is sufficient to prevent posterior collapse. Shen et al. (2021) extended this method with dropout on the variances of posteriors to learn a more diverse and less uncertain latent space.

3. Scale-VAE

3.1. The Proposed Method

As discussed above, since the information about input data in the latent variables is difficult to exploit, VAE benefits more easily from only relying on the expressive decoder to reconstruct the data. In the initial stages of training, there is a lot of noise in the latent variables, which makes it difficult for the model to use the information in them. As the training progresses, under the pressure of the KL term, the posterior distributions of different data instances become so similar that the model cannot distinguish effectively. So, the model abandons the latent variables, causing posterior collapse.

Intuitively, if the difficulty of obtaining information from latent variables can be reduced, the model may be willing to take advantage of the latent variables and thus jump out of the degenerate local optimum at the time of posterior collapse to achieve better results. Furthermore, a VAE without posterior collapse is not necessarily a good model, in addition to learning effective representations, the model should also be able to generate high-quality results and learn a continuous and semantically meaningful latent space.

Motivated by this, we propose Scale-VAE to achieve the following three goals: making the distinction between the posteriors of different data instances more significant to prevent posterior collapse; learning a smooth latent space to guarantee the generation quality; maintaining semantic consistency and coherence of the learned latent space.

Given a data instance $x \in X$, the encoder $Enc_\phi(\cdot)$ parametrizes the posterior distribution $q_\phi(z|x)$ as a n-dimensional Gaussian distribution $N(\mu_x, \sigma_x^2)$ with mean $\mu_x = (\mu_{x,1}, \mu_{x,2}, ..., \mu_{x,n})$ and diagonal covariance σ_x^2. Using $Std[\mu_{X,d}]$ to denote the standard deviation for the dth dimension of the mean μ_x across the data instances X, we define the scale-up factor as:

$$f = (f_1,\, f_2,\, ...,\, f_d,\, ...,\, f_n), \tag{2}$$

$$f_d = \frac{des\_std}{Std[\mu_{X,d}]}, \tag{3}$$

where f is a n-dimensional vector, f_d is the dth dimension, and des_std is a parameter for adjusting the standard deviation.

The factor f is used to derive a scaled-up posterior distribution:

$$\hat{q}_\phi(z|x) = N(\hat{\mu}_x,\, \sigma_x^2), \tag{4}$$

$$\hat{\mu}_x = (\hat{\mu}_{x,1},\, \hat{\mu}_{x,2},\, ...,\, \hat{\mu}_{x,d},\, ...,\, \hat{\mu}_{x,n}), \tag{5}$$

$$\hat{\mu}_{x,d} = f_d \cdot \mu_{x,d}. \tag{6}$$

Then the objective to optimize in the framework of Scale-VAE is:

$$\begin{aligned} L_{scale} = E_{p(X)}[E_{\boldsymbol{\hat{q}_\phi(z|x)}}[\log p_\theta(x|z)] \\ - D_{KL}[q_\phi(z|x)\|p(z)]]. \end{aligned} \tag{7}$$

In contrast to the objective of VAE in Eq. 1, the objective of Scale-VAE in Eq. 7 merely replaces $q_\phi(z|x)$ in the reconstruction term with the scaled-up posterior $\hat{q}_\phi(z|x)$. Under the distributions $\hat{q}_\phi(z|x) = N(\hat{\mu}_x,\, \sigma_x^2)$, the standard deviation $Std[\hat{\mu}_{X,d}]$ for each dimension of the mean $\hat{\mu}_x$ across the data instances X is des_std. By setting a appropriate parameter des_std, Scale-VAE can distinguish the latent variables of different data instances and obtain effective information from them more easily.

The KL term in the objective of Scale-VAE is the same as for VAE. There are three considerations for doing so. First, it prevents the model from pushing the posterior $q_\phi(z|x)$ even further into the prior $p(z)$. Second, the model does not force the KL term to be larger than a certain positive constant, but aims to make the whole objective reach the optimum. Third, during inference stage, after sampling a latent variable z from the prior $p(z)$, we multiply it by the factor f and then input it to the decoder $Dec_\theta(\cdot)$ for generation, which is consistent with the training stage. The latent space used for generation is smooth without many discontinuous holes due to large standard deviation of $\hat{\mu}_x$.

Typically, the model is trained in mini-batches. The factor f in Eq. 2 and Eq. 3 can only be computed within a mini-batch. But in this case, the factors used in each mini-batch are different, which will cause clutter in the latent space. To solve this problem, each mini-batch uses its own factor f only in the initial f_epo training epochs, and thereafter we record the factors f of all the mini-batches in each training epoch and then take the average $\bar{f}$ as the factor of the next training epoch.

Algorithm 1 shows the training procedure of Scale-VAE.

Algorithm 1 Training Procedure of Scale-VAE

```
Initialize $\phi$, $\theta$, $des\_std$ and $f\_epo$
$i \leftarrow 1$
while not convergence do
    for $\mathbf{x}$ in mini-batches do
        $\mu_{\mathbf{x}},\, \sigma_{\mathbf{x}}^2 = Enc_\phi(\mathbf{x})$
        $f = des\_std / Std[\mu_{\mathbf{x}}]$
        if $i \leq f\_epo$ then
            $\hat{\mu}_{\mathbf{x}} = f \cdot \mu_{\mathbf{x}}$
        else
            $\hat{\mu}_{\mathbf{x}} = \bar{f} \cdot \mu_{\mathbf{x}}$
        end if
        Sample $z \sim N(\hat{\mu}_{\mathbf{x}},\, \sigma_{\mathbf{x}}^2)$
        Generate $\mathbf{x}$ from $Dec_\theta(z)$
        $g_{\phi,\,\theta} \leftarrow -\nabla_{\phi,\,\theta}\, L_{scale}(\mathbf{x};\, \phi,\, \theta)$
        Update $\phi$, $\theta$ according to $g_{\phi,\,\theta}$
    end for
    $\bar{f} = Average(f)$
    $i \leftarrow i + 1$
end while
```

3.2. Connections with Previous Work

To measure the diversity of posteriors, Ma et al. (2019) proposed the mutual posterior diversity (MPD) that is defined as:

$$MPD = E_{p(X)}[D_{KL}[q_\phi(z|x_1)\|q_\phi(z|x_2)]], \tag{8}$$

where $x_1,\, x_2 \sim P(X)$ are $i.i.d.$ data instances. Shen et al. (2021) refined it with symmetric KL divergence, and it can be computed in an analytical way under Gaussian distributions:

$$\begin{aligned} 2MPD &= 2E_{p(X)}[D_{SKL}[q_\phi(z|x_1)\|q_\phi(z|x_2)]] \\ &= \sum_{d=1}^{n} E_{p(X)}[\frac{(\mu_{x_1,d} - \mu_{x_2,d})^2}{\sigma_{x_1,d}^2}] + \\ &\quad \sum_{d=1}^{n} E_{p(X)}[\sigma_{x,d}^2] E_{p(X)}[\frac{1}{\sigma_{x,d}^2}] - 1. \end{aligned} \tag{9}$$

If $\sigma_{x,d}^2$ is upper bounded, MPD has one lower and strict bound:

$$MPD \geq \frac{1}{C} \sum_{d=1}^{n} Var_{p(X)}[\mu_{x,d}], \text{ if } \sigma_{x,d}^2 \leq C, \tag{10}$$

for which a detailed proof can be found in Shen et al. (2021). In Scale-VAE, the value of $\sigma_{x,d}^2$ is around 1 under the constraint of the KL term. So, the Scale-VAE method is equivalent to increasing the MPD, that is, increasing the divergence among the posterior distribution family for better distinguish.

MPD can also be written as follows (Ma et al., 2019):

$$\begin{aligned} MPD = E_{p(X)}[D_{KL}[q_\phi(z|x)\|q_\phi(z)] \\ + D_{KL}[q_\phi(z)\|q_\phi(z|x)]], \end{aligned} \tag{11}$$

where $q_\phi(z) = E_{p(X)}[q_\phi(z|x)]$ is the aggregated posterior.

We now come to the MI (Alemi et al., 2016) between input instances x and latent variables z under the distribution $q_\phi(x, z)$:

$$\begin{aligned} I_q(x; z) =& E_{p(X)}[D_{KL}[q_\phi(z|x) \| p(z)]] \\ & - D_{KL}[q_\phi(z) \| p(z)] \end{aligned} \tag{12}$$

$$= E_{p(X)}[D_{KL}[q_\phi(z|x) \| q_\phi(z)]]. \tag{13}$$

According to Eq. 11 and Eq. 13, MPD is a symmetric version of the KL-divergence in MI. So, the Scale-VAE method is also equivalent to increasing the MI between latent variables and input data.

Scale-VAE has some commonalities with the batch normalization-based method (Zhu et al., 2020; Shen et al., 2021). Batch normalized-VAE (BN-VAE) regularizes $\mu_{x,d}$ with batch normalization:

$$\hat{\mu}_{x,d} = \gamma \frac{\mu_{x,d} - \mu_{Bd}}{\sigma_{Bd}} + \beta, \tag{14}$$

where μ_{Bd} and σ_{Bd} denote the mean and standard deviations of $\mu_{x,d}$ in a mini-batch. After this regularization, the distribution of $\mu_{x,d}$ has the mean of β and the standard deviation of γ, and the expectation of KL has a lower bound $E[KL] \geq n \cdot (\gamma^2 + \beta^2)/2$.

However, keeping the expectation of KL term positive is not a sufficient condition to avoid posterior collapse. For example, if we set $\gamma = 0, \beta > 0$, $E[KL]$ has a positive lower bound, but posterior collapse still occurs. In addition, the regularization is performed within each mini-batch, which results in different degrees of scaling on the posteriors.

Compared to BN-VAE, Scale-VAE has the following differences. First, the motivation is not to force the KL term to be larger than a certain positive constant, but to use the scaled-up posteriors for generation, thus making it easier for the model to exploit the information in latent variables. Second, all data instances use the same scale-up factor, preventing clutter in the latent space. Third, the KL term is still computed using the original posteriors; the model can feel free to optimize the KL term, and the scaling-up will help the model make better use of the latent variables; this alleviates the contradiction between the reconstruction term and the KL term, so that the model can not only learn the latent space that is smooth and tends to the prior, but also improve the density estimation.

4. Experiments

4.1. Experimental Setup

Configurations. Both the encoder and decoder are implemented with a one-layer LSTM. The hidden size is 1024. For all experiments we used a Gaussian prior $N(0, I)$. An affine transformation of the 32-dimensional latent variable z is used as the initial hidden state of the decoder, and z is also concatenated with input for the decoder. The dimension of word embedding layer is 512. The LSTM layer and embedding layer are initialized with uniform distributions on $[-0.01, 0.01]$ and $[0.1, 0.1]$, respectively. Dropout layers with probability 0.5 are applied to both the word embeddings and the last output features of the decoder. We utilized the SGD optimizer with 32 data instances per mini-batch and started with a learning rate of 1.0 and decayed it by 0.5 if the validation loss has not improved in 5 epochs. We stopped training after 5 learning rate decays with the maximum number of epochs as 120. We applied a linear annealing strategy to increase the KL weight from 0 to 1 in the first 10 epochs.

Baselines. We compared Scale-VAE with various models including those holding the state-of-the-art performance on text modeling benchmarks: the LSTM language model (**LSTM-LM**); VAE with annealing (Bowman et al., 2016); **VAE** using default **cyclical** annealing schedule in Fu et al. (2019); ***β*-VAE** (Higgins et al., 2016) using parameter β to re-weight the KL term; **Free-Bits** (Kingma et al., 2016) that stops the optimization of KL term when it is smaller than a threshold; ***δ*-VAE** (Razavi et al., 2019) using a parameter to constrain the minimum of KL term by setting the mean and variance of posteriors in a specific range; **SA-VAE** (Kim et al., 2018) combining amortized variational inference and stochastic variational inference; **Skip-VAE** (Dieng et al., 2019) that adds direct paths between latent variables and hidden layers of the decoder; **Agg-VAE** (He et al., 2018) that trains the encoder aggressively; **MAE** (Ma et al., 2019) using two parameters to control the diversity and smoothness of the latent space; **BN-VAE** (Zhu et al., 2020) using batch normalization to control the lower bound on the expectation of KL term; **DU-VAE** (Shen et al., 2021) using batch normalization and variance dropout to control the diversity and uncertainty of the latent space.

4.2. Density Estimation

We conducted experiments on two benchmark datasets: Yahoo and Yelp corpora (Yang et al., 2017). We evaluated our method with the approximate negative log-likelihood (NLL) estimated by 500 importance weighted samples (Burda et al., 2016). It provides a tighter lower bound compared to ELBO and shares the same information with perplexity (PPL). We reported the value of the KL term $D_{KL}[q_\phi(z|x) \| p(z)]$. We also computed the MI $I_q(x; z)$ (Alemi et al., 2016) as described in Eq. 13 under the distribution $q_\phi(x, z)$[1] and the number of

[1] For Scale-VAE, $I_q(x; z)$ is computed under the scaled-up distribution $\hat{q}_\phi(x, z)$.

	Yahoo				Yelp			
Model	NLL	KL	MI	AU	NLL	KL	MI	AU
LSTM-LM	328.0	-	-	-	357.5	-	-	-
VAE	328.6	0.2	0.2	0.8	358.0	0.1	0.1	0.2
cyclical	330.5	2.1	2.1	2.3	359.3	2.1	2.0	4.2
β-VAE(0.4)	328.4	7.7	7.1	7.3	358.0	5.4	5.1	3.6
Free-Bits(0.1)	328.3	3.4	2.4	32.0	357.0	4.8	2.6	32.0
δ-VAE(0.1)	329.7	3.2	0.0	2.0	357.9	3.2	0.0	0.0
SA-VAE$*$	327.2	5.2	3.7	9.8	355.9	2.8	1.7	8.4
Skip-VAE$*$	328.5	2.3	1.3	8.1	357.6	1.9	1.0	7.4
Agg-VAE$*$	326.7	5.7	2.9	15.0	355.9	3.8	2.4	11.3
MAE$*$(1, 0.2|2, 0.2)	332.1	5.8	3.5	28.0	362.8	8.0	4.6	32.0
BN-VAE(0.6)	326.9	6.5	5.8	32.0	356.6	6.5	5.7	32.0
BN-VAE(0.0, 0.7)	331.3	7.8	0.0	0.0	360.4	7.8	0.0	0.0
DU-VAE(0.6, 0.8|0.5, 0.8)	326.9	8.8	7.2	28.0	356.2	6.7	5.9	20.0
Scale-VAE(0.7, 1|0.7, 7)	325.0	7.1	8.3	32.0	353.7	5.4	8.2	32.0
Scale-VAE(0.9, 1|0.9, 7)	323.1	9.5	9.1	32.0	351.7	7.2	9.1	32.0
Scale-VAE(1.1, 1|1.1, 7)	**321.3**	8.4	9.2	32.0	**349.9**	6.7	9.2	32.0
Scale-VAE$\star$(0.7, 1|1.1, 7)	325.9	6.4	7.9	32.0	350.7	8.1	9.2	32.0

Table 1: Density estimation performance on Yahoo and Yelp. The results are the mean values across 5 different random runs. $*$ indicates the results are referred from Kim et al. (2018), He et al. (2018) and Shen et al. (2021). $\star$ indicates that KL annealing is not used. Hyperparameters are listed in brackets and split by | if different on different datasets.

activate units (AU) (Burda et al., 2016) in the latent space. The activity of a latent dimension d is measured as $A_{z_d} = Cov_{\mathbf{x}}(E_{z_d \sim q(z_d|\mathbf{x})}[z_d])$. If $A_{z_d} > 0.01$, the dimension d is regarded as active.

Table 1 shows the results[2]. We use NLL as the main metric in this part because it is the most direct metric to evaluate the performance of density estimation. Moreover, both the reconstruction and the KL values are included in NLL, and this combined value is important for VAE that needs to make a trade-off between the reconstruction term and the KL term. All Scale-VAE models with different des_std and f_epo achieve better NLL than previous state-of-the-art models on both datasets. We also experimented with Scale-VAE without KL annealing, and the results show that it still does not suffer from posterior collapse and achieves a better NLL than previous models.

Scale-VAE also achieves the best results on KL, MI, and AU, but we do not take the three metrics as the most important criteria to judge the best model. When the VAE suffers from posterior collapse, the KL term usually tends toward 0, but a relatively high KL does not guarantee that the VAE does not have posterior collapse. For example, in BN-VAE, if setting $\gamma = 0.0$ and $\beta = 0.7$, we got a KL value around $n\beta^2/2 = 7.8$, but posterior collapse still occurs, and both MI and AU are zero, because the mean values of posterior distributions for different data instances are all β. So, it is important that the different data instances have a proper degree of discrimination in the latent space, rather than a higher KL. A higher MI does indicate that there is more information about the input data in the latent variables. However, the model can obtain high MI by making different data instances as far away from each other as possible in the latent space, but this will cause the latent space to become unsmooth. Therefore, MI alone is also not enough to prove the advantages of a model. For example, β-VAE has a higher MI but a worse NLL than BN-VAE. A higher AU indicates that more dimensions are active in the latent variables, that is, discriminative to the data instances. However, further evaluation is needed to determine whether there is redundancy among different dimensions. For example, Free-Bits has a higher AU but a worse NLL than SA-VAE.

An advantage of BN-VAE is that it can solve posterior collapse while maintaining a training cost similar to VAE. Table 2 shows the comparison of the training time to convergence of VAE, BN-VAE, and Scale-VAE. The training cost of Scale-VAE is almost the same as or even lower than that of basic VAE and BN-VAE, because it only performs a scale-up operation on the mean of the approximate posterior, and this operation accelerates the model to obtain information from latent variables.

[2] When initializing the LSTM layer and embedding layer in DU-VAE with uniform distributions on $[-0.01, 0.01]$ and $[0.1, 0.1]$, we did not get usable results because the KL term is extremely large. Therefore, for DU-VAE, we used the default initial values for LSTM layers and embedding layers in PyTorch (v1.12). To make comparisons in the same configurations, we selected BN-VAE as the representative of batch normalization-based methods in subsequent experiments.

	Yahoo		Yelp	
Model	Hours	Ratio	Hours	Ratio
VAE	3.50	1.00	4.90	1.00
BN-VAE	3.55	1.01	4.70	0.95
Scale-VAE	**3.22**	**0.91**	**4.29**	**0.87**

Table 2: Comparison of training time to convergence. Ratio indicates the relative ratio to VAE. The results are the mean values across 5 different random runs with the best parameters in Table 1 for each model.

Now let's discuss the effect of parameters des_std and f_epo on Scale-VAE. des_std is to control the degree of discrimination among the scaled-up posterior distributions and f_epo is to prevent clutter in the latent space. In the initial f_epo training epochs, the standard deviation of the mean values for the posterior distributions $Std[\mu_{\mathbf{x}}]$ varies greatly, so the average factor $\bar{f}$ from the last epoch cannot guarantee that $Std[\hat{\mu}_{\mathbf{x}}]$ in this epoch is des_std. After f_epo epochs, $Std[\mu_{\mathbf{x}}]$ tends to be stable, then $\bar{f}$ of the last epoch can be used to keep $Std[\hat{\mu}_{\mathbf{x}}]$ around des_std, avoiding clutter in the latent space.

Yelp dataset requires a larger f_epo than Yahoo because the average length of data in Yelp is longer and $Std[\mu_{\mathbf{x}}]$ in the initial stages of training varies more. We used $f\_epo = 7$ for Yelp and $f\_epo = 1$ for Yahoo. A larger des_std requires a larger f_epo because it makes the factor $\bar{f}$ more sensitive to the changes in $Std[\mu_{\mathbf{x}}]$. As shown in Table 3, for Yahoo dataset, when des_std is $0.3, 0.5, 0.7, ..., 1.1$, a f_epo of 1 is sufficient, but when des_std is 1.3, f_epo needs to be set to 3. However, when f_epo is already sufficient to ensure that the model can pass the first few epochs with large changes in $Std[\mu_{\mathbf{x}}]$, increasing f_epo further does not have a significant impact on the results. For example, for Yahoo, when des_std is 0.7, setting f_epo to 3 or 5 will yield results similar to $f\_epo = 1$.

In general, NLL will first decrease and then increase with the increase of des_std. As shown in Table 3, on Yahoo, the NLL values are $325.7,\ 325.0,\ 323.1,\ 321.3,\ 318.6$ for des_std of $0.5,\ 0.7,\ 0.9,\ 1.1,\ 1.3$, but when des_std is 1.5, NLL is 326.5 and unstable. The possible reason is that although a large des_std makes different data instances more discriminative, it also affects the model to establish connections between different data instances. When des_std is 0.3, the model gets good NLL but not very good AU, and the KL term is relatively small, so the reconstruction term may not be very good compared to the case of similar NLL but large KL term like $des\_std = 0.7$.

Scale-VAE		Yahoo			
des_std	f_epo	NLL	KL	MI	AU
0.3	1	324.1	1.8	8.4	23.2
0.5	1	325.7	3.7	7.0	31.8
0.7	1	325.0	7.1	8.3	32.0
0.7	3	324.6	6.9	8.4	32.0
0.7	5	324.7	7.1	8.3	32.0
0.9	1	323.1	9.5	9.1	32.0
1.1	1	321.3	8.4	9.2	32.0
1.3	3	318.6	8.8	9.2	32.0
1.5	7	326.5	3.0	8.3	32.0

Table 3: Density estimation performance of Scale-VAE with different hyperparameters on Yahoo and Yelp. The results are the mean values across 5 different random runs.

#labeled-data	100	500	1K	2K	10K
VAE	72.0	75.9	76.5	78.6	80.0
β-VAE(0.4)	82.0	83.7	84.3	84.8	86.2
FB(0.1)	72.0	75.9	76.5	78.6	80.0
δ-VAE(0.1)	58.9	59.8	60.5	59.7	61.2
Agg-VAE$*$	75.1	77.2	78.5	79.3	80.1
MAE$*$(2, 0.2)	61.5	61.7	62.4	63.6	63.7
BN-VAE(0.6)	85.4	88.7	89.8	90.2	90.4
DU-VAE(0.5, 0.8)	85.1	86.4	88.2	89.0	89.1
DU-VAE$*$(0.5, 0.8)	**88.9**	89.6	90.4	90.5	90.8
Scale-VAE(0.7, 1)	87.7	**89.8**	**90.7**	**91.3**	**91.2**

Table 4: Classification accuracy with different amounts of labeled data in Yelp. $*$ indicates the results are referred from Shen et al. (2021). Hyperparameters are listed in brackets.

4.3. Representation Learning

We evaluated the quality of the learned representations by training a one-layer linear classifier using the means of posterior distributions. We conducted experiments on a downsampled version of Yelp sentiment dataset (Shen et al., 2017). Table 4 shows the classification accuracy with different amounts of labeled data. We did not reproduce the best results of DU-VAE, so both our experimental result and that in Shen et al. (2021) are listed. In the case of 100 labeled data, the result from Shen et al. (2021) has the highest classification accuracy. Scale-VAE achieves the highest classification accuracy with all other amounts of labeled data.

4.4. Latent Space Property

As discussed in Section 3.2, in batch normalization-based methods, the regularization is performed within each mini-batch, which results in different degrees of scaling on the posteriors. We argued that this may create clutter in the latent space. Scale-VAE can avoid this problem. To verify this, we conducted an intuitive comparison of latent spaces

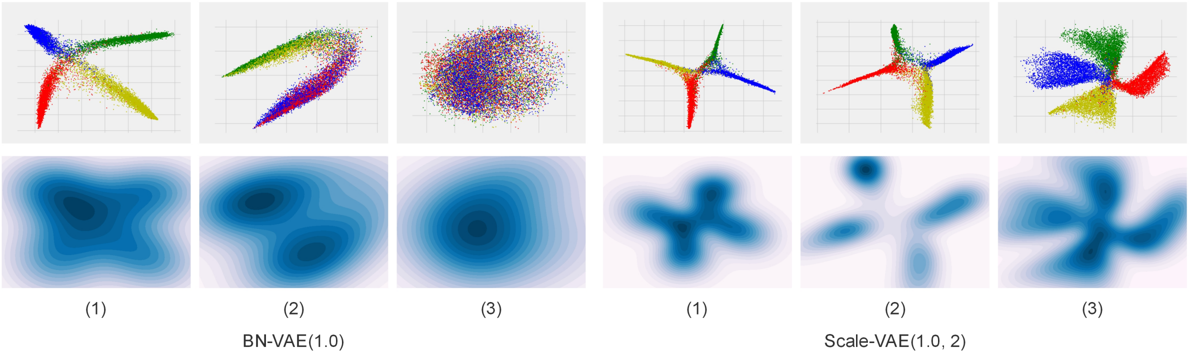

Figure 1: The visualization of the latent spaces learned by BN-VAE and Scale-VAE. The first row shows the means of the posteriors for data instances generated by different Gaussian components in different colors. The second row shows the contour plot of the aggregated posterior. The darker the color, the higher the probability. (1), (2), and (3) correspond to the three modes of dividing the data instances into mini-batches.

learned by BN-VAE and Scale-VAE on a synthetic dataset.

Following He et al. (2018), we sampled 2-dimensional latent variables z from a Gaussian mixture distribution that has four mixture components with mean values of $(-2.0, -2.0)$, $(-2.0, 2.0)$, $(2.0, -2.0)$, $(2.0, 2.0)$ and unit variance. Then we generated synthetic data instances from an one-layer LSTM conditioned on these latent variables. The hidden size is 100. An affine transformation of the latent variable z is used as the initial hidden state of the decoder, and z is also concatenated with the output of LSTM at each time stamp to be directly mapped to vocabulary space. The dimension of input embedding layer is 100. The LSTM layers and the layer mapping z to vocabulary are initialized with uniform distribution on $[-1, 1]$ and $[-5, 5]$, respectively. We generated a dataset containing $20K$ instances of length 10 from a vocabulary of size 1000 and divided it into training/validation/test sets with ratio of $8/1/1$.

In the synthetic experiments, both the encoder and decoder are implemented with a one-layer LSTM. The hidden size is 50. The dimensions of latent variables z and embedding layers are 2 and 50, respectively. The other experimental configurations are the same as those in Section 4.1.

We used three modes to divide the data instances into mini-batches. First, the data instances generated by different Gaussian components are randomly distributed in different mini-batches, and the division of mini-batches remains constant across all training epochs. Second, the data instances generated by different Gaussian components are randomly distributed in different mini-batches, and the mini-batches are repartitioned at each training epoch. Third, the data instances in the same mini-batch come from the same Gaussian component, and the division of the mini-batches remains constant across all training epochs. To visualize the learned latent spaces, we plotted the mean of the approximate posterior for different data instances x and the contour plot of the aggregated posterior $q_\phi(z) = E_{p(X)}[q_\phi(z|x)]$[3].

Figure 1 shows the results. Scale-VAE can distinguish data instances generated by four Gaussian components in all three mini-batch partitioning modes. However, BN-VAE can only work under the first partitioning mode. In the second mode, BN-VAE cannot distinguish the data instances clearly, and in the third extreme mode, the latent space learned by BN-VAE is a mess. This verifies our view that the batch normalization-based methods cause clutter in the latent space.

The aggregated posterior for Scale-VAE in Figure 1 is the scaled-up posterior $\hat{q}_\phi(z) = E_{p(X)}[\hat{q}_\phi(z|x)]$. In Scale-VAE, the scaled-up posteriors are used in subsequent decoding, and the means of these posteriors are the learned representations for data instances, but the original posteriors are still used to optimize the KL term in the objective. During inference stage, when a latent variable z is sampled from the prior $p(z)$, it needs to be multiplied by the factor f first before being fed to the decoder for generation, which is consistent with the training stage. Intuitively, to learn a generative model well, the divergence between the original aggregated posterior and the prior should be kept small. To evaluate this point, under the first mini-batch partitioning mode for the synthetic dataset, we plotted the contour plots of the prior $N(0, I)$, the aggregated posterior in BN-VAE, and

[3] For Scale-VAE, the aggregated posterior is computed using the scaled-up posteriors $\hat{q}_\phi(z) = E_{p(X)}[\hat{q}_\phi(z|x)]$. Compared to BN-VAE, the contour plot of Scale-VAE covers a larger area. We have scaled it down for better comparison.

the original aggregated posterior in Scale-VAE. Figure 2 shows the results. It can be seen that the aggregated posterior distribution of Scale-VAE is closer to the prior distribution than that of BN-VAE.

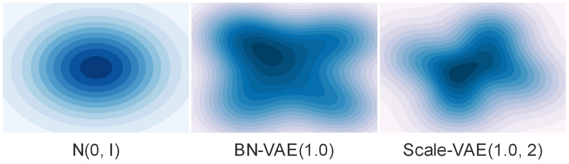

Figure 2: The contour plots of the prior N(0, I), the aggregated posterior in BN-VAE, and the original aggregated posterior in Scale-VAE. All plots are located in the same region ([-2, 2] × [-2, 2]) with the same scale.

To further evaluate the smoothness of the latent space and the quality and diversity of the generated sentences, we conducted the experiments as follows. Firstly, two latent variables $z1$ and $z2$ are obtained by sampling from the prior distribution $N(0, I)$. Then, linear interpolation is performed between $z1$ and $z2$ to obtain 5 latent variables. After that, the above steps are repeated 1000 times, resulting in 5000 latent variables. Finally, these latent variables are used for greedy decoding, resulting in 5000 sentences. The maximum sentence length is set to 15. We conducted the above process with 10 random seeds using BN-VAE and Scale-VAE trained on Yahoo in Section 4.2.

To evaluate the quality of the generated sentences, we computed PPL of the sentences using the LSTM-LM trained on Yahoo and calculated the proportion of 2-grams, 3-grams, and 4-grams in the sentences that appear in the Yahoo dataset. To evaluate the diversity of the generated sentences, we calculated the entropy of the sentences and the proportion of different 1-grams, 2-grams, 3-grams in the generated sentences. Table 5 shows the results. Except for the proportion of 2-grams, BN-VAE gets better results than Scale-VAE on other quality metrics. We suspect there are two possible reasons: First, the diversity of sentences generated by Scale-VAE is much better than that of BN-VAE as shown in Table 5. The higher the diversity, the greater the probability of error. Second, Scale-VAE scales up the latent variables, so for the same latent space, there will be more coordinate points containing different meaningful information. Taking the 1-dimensional space as an example, if the interval is $[0, 1]$, BN-VAE divides it into $(0, 0.1, 0.2, ..., 0.9, 1)$, but Scale-VAE may divide it into $(0, 0.01, 0.02, 0.03, ..., 0.98, 0.99, 1)$, which may require more data instances to train the model.

Table 6 shows some generated sentences from the prior distribution on the downsampled version of Yelp dataset. The middle sentences were generated using the interpolation between the latent variables of sentences at both ends, each word token was obtained using probability sampling.

Model	BN-VAE(0.6)	Scale-VAE(1.1, 1)
PPL	**260.44**	274.93
Gram-2	97.77	**98.08**
Gram-3	**87.24**	84.19
Gram-4	**68.10**	60.24
Entropy	5.72	**5.91**
Dist-1(E-02)	0.82	**1.33**
Dist-2(E-05)	0.90	**1.89**
Dist-3(E-05)	1.69	**4.40**

Table 5: Quality and diversity evaluation results of the generated sentences on Yahoo. Gram-2/3/4 denote the proportion of 2/3/4-grams in the generated sentences that appear in the Yahoo dataset. Dist-1/2/3 denote the proportion of different 1/2/3-grams in the generated sentences.

the owner is a friendly waitress and the food is amazing!
great selection and the oven specials are good!
the prices are great and the people are superb!
the rolls are really delicious!
the chips are crispy and delicious!
the staff is friendly and a few regulars.
the sales team is fast and good!
the bartenders are friendly , it 's a treat.
the employees are friendly and extremely helpful.
the prices are fair and everyone is a happy customer.

Table 6: Sentences generated from the prior distribution on Yelp.

5. Conclusion

In this paper, we focus on the posterior collapse problem in VAE on text modeling when employing a autoregressive decoder. Starting from the analysis of the causes, we summarize several directions for solving the posterior collapse problem. Motivated by reducing the difficulty of exploiting latent variables by the decoder, we propose Scale-VAE. Instead of forcing the KL term to be larger than a certain positive constant, Scale-VAE enables the decoder to make full use of the information in latent variables, so as to freely balance the optimization of the reconstruction term and the KL term, and jump out of the degenerate local optimum when posterior collapse occurs. We compare Scale-VAE with state-of-the-art models. Experimental results show that Scale-VAE gains the best result in density estimation and representation learning. It does not cause clutter in the latent space and is able to generate high-quality diverse sentences from the smooth latent space.

Acknowledgments

This work was supported by Beijing Natural Science Foundation (No. 4234081), International Chinese Language Education Research Program (No. 23YH35C), and National Natural Science Foundation of China (No. 62177001).

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. In *International Conference on Learning Representations*, pages 1–19.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *International Conference on Learning Representations*, pages 1–14.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. In *International Conference on Learning Representations*, pages 1–17.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational autoencoders. In *Conference on Uncertainty in Artificial Intelligence*, pages 856–865.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2019. Avoiding latent variable collapse with generative skip models. In *International Conference on Artificial Intelligence and Statistics*, pages 2397–2405.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 240–250.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Serhii Havrylov and Ivan Titov. 2020. Preventing posterior collapse with levenshtein variational autoencoder. *arXiv preprint arXiv:2004.14758*, pages 1–14.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*, pages 1–15.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. Beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, pages 1–22.

Sepp Hochreiter and Jürgen Schmidhuber. 2012. Long short-term memory. *Neural computation*, pages 37–45.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*, pages 1–14.

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 1–16.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*, pages 3603–3614.

Xuezhe Ma, Chunting Zhou, and Eduard Hovy. 2019. Mae: Mutual posterior-divergence regularization for variational autoencoders. In *International Conference on Learning Representations*, pages 1–16.

Tom Pelsmaeker and Wilker Aziz. 2020. Effective estimation of deep generative language models. In *Annual Meeting of the Association for Computational Linguistics*, pages 7220–7236.

Alban Petit and Caio Filippo Corro. 2021. Preventing posterior collapse in variational autoencoders for text generation via decoder regularization. In

NeurIPS Workshop on Deep Generative Models and Downstream Applications, pages 1–7.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing posterior collapse with delta-vaes. In *International Conference on Learning Representations*, pages 1–24.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1–12.

Dazhong Shen, Chuan Qin, Chao Wang, Hengshu Zhu, Enhong Chen, and Hui Xiong. 2021. Regularizing variational autoencoder with diversity and uncertainty awareness. In *International Joint Conference on Artificial Intelligence*, pages 2964–2970.

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 1–12.

Tianbao Song, Jingbo Sun, Xin Liu, Jihua Song, and Weiming Peng. 2022. Topic-word-constrained sentence generation with variational autoencoder. *Pattern Recognition Letters*, 160:148–154.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798.

Aaron Van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 1–10.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, pages 3881–3890.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. Infovae: Balancing learning and inference in variational autoencoders. In *AAAI conference on Artificial Intelligence*, pages 5885–5892.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Annual Meeting of the Association for Computational Linguistics*, pages 1098–1107.

Huangjie Zheng, Jiangchao Yao, Ya Zhang, Ivor W Tsang, and Jia Wang. 2019. Understanding vaes in fisher-shannon plane. In *AAAI conference on Artificial Intelligence*, pages 5917–5924.

Qile Zhu, Wei Bi, Xiaojiang Liu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. A batch normalized inference network keeps the kl vanishing away. In *Annual Meeting of the Association for Computational Linguistics*, pages 2636–2649.