Recurrence Boosts Diversity! Revisiting Recurrent Latent Variable in Transformer-Based Variational AutoEncoder for Diverse Text Generation

Jinyi Hu[1,2,3], **Xiaoyuan Yi**[5], **Wenhao Li**[1,2,3], **Maosong Sun**[1,2,3,4*], **Xing Xie**[5]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing
[2] Beijing National Research Center for Information Science and Technology
[3] Institute for Artificial Intelligence, Tsinghua University, Beijing
[4] Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou
[5] Microsoft Research Asia
hu-jy21@mails.tsinghua.edu.cn, xiaoyuanyi@microsoft.com

Abstract

Variational Auto-Encoder (VAE) has been widely adopted in text generation. Among many variants, recurrent VAE learns token-wise latent variables with each conditioned on the preceding ones, which captures sequential variability better in the era of RNN. However, it is unclear how to incorporate such recurrent dynamics into the recently dominant Transformer due to its parallelism. In this work, we propose TRACE, a Transformer-based recurrent VAE structure. TRACE imposes recurrence on segment-wise latent variables with arbitrarily separated text segments and constructs the posterior distribution with residual parameterization. Besides, we design an acceleration method by approximating idempotent matrices, which allows parallelism while maintaining the conditional dependence of latent variables. We demonstrate that TRACE could enhance the entanglement of each segment and preceding latent variables and deduce a non-zero lower bound of the KL term, providing a theoretical guarantee of generation diversity. Experiments on two unconditional and one conditional generation tasks show that TRACE achieves significantly improved diversity while maintaining satisfactory generation quality.

1 Introduction

Variational Auto-Encoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) has thrived in various text generation tasks due to its ability to learn flexible representations, such as machine translation (Shah and Barber, 2018) and the generation of dialogue (Zhao et al., 2017), story (Yu et al., 2020) and poetry (Yi et al., 2020). To further improve expressive power, diverse VAE variants have been proposed. For example, IWAE (Burda et al., 2016) optimizes a tighter lower bound; Ladder VAE (Sønderby et al., 2016) learns hierarchical latent representations and vMF-VAE (Xu and Dur-

rett, 2018) replaces Gaussian distributions with von Mises-Fisher distributions.

Among all variants, temporal VAE (Fabius et al., 2015; Chung et al., 2015) was prevalent in the era of RNN, which captures temporal variability by introducing the dependency of a series of latent variables with each associated with one time step. Such a VAE variant has succeeded in kinds of sequence modeling tasks, *e.g.*, dialog generation (Kim et al., 2020), audio generation (Franceschi et al., 2020), and time series prediction (Li et al., 2019).

Temporal VAE can be categorized into three paradigms according to the dependency of prior distributions at each time step: a) independent normal distributions (abbr. *IND*) (Li et al., 2020c), b) context-conditioned Gaussian distributions (abbr. *CGD*) (Du et al., 2018) which are conditioned on preceding text, and c) recurrent Gaussian distributions (abbr. *RGD*), *i.e.*, Recurrent VAE (Chung et al., 2015), which are conditioned on preceding both text and latent variables[1]. Both IND and CGD ignore the interaction of latent variables, limiting their expressive ability. In comparison, by introducing the dependency of latent variables, RGD could better model the sequential variability and thus greatly improve generation diversity while maintaining satisfactory quality. We provide the theoretical proof of such an advantage in Sec. 4.3.

These paradigms can be easily implemented with RNN benefiting from RNN's natural recurrent structure. Stepping into the age of Transformer (Vaswani et al., 2017), it is promising to adapt temporal VAE to this popular architecture. *IND* and *CGD* paradigms are naturally compatible with Transformer because their latent variables at each time step are independent which could be simply combined with the parallel computation of Transformer self-attention via causal and noncausal masks (Lin et al., 2020). However, there are no off-the-shelf solutions to incorporate *RGD* into

---

[1]See Sec. 3.2 for mathematical details of these paradigms.

Transformer-based VAEs, since recurrence would be a natural obstacle to parallelism (recurrent latent variables need to be sequentially sampled), which limits the capacity of this potential VAE paradigm.

Could we equip Transformer with such recurrent dynamics for better diversity while keeping the training parallelism? To answer this question, we propose TRACE[2], a novel Transformer-based recurrent VAE structure. TRACE imposes recurrence on segment-wise (instead of token-wise) latent variables with arbitrary segmentation, *e.g.*, sentences or segments with specified length. Besides, we construct the posterior distribution using residual parameterization and layer normalization, which could deduce a non-zero lower bound of the KL loss to alleviate KL vanishing (Bowman et al., 2016). Moreover, to accelerate training, we design a method to recover the parallelism in Transformer by approximating idempotent parameter matrices for the latent space, leading to improved diversity, satisfactory quality, and faster training.

In summary, our contributions are as follows: We are the first to (i) incorporate recurrent VAE into Transformer with recurrence on segment-wise latent variables which allows a flexible trade-off of diversity and quality; (ii) propose a method to recover parallelism and accelerate training with comparable performance; (iii) mathematically demonstrate that our model has a deducted non-zero lower bound to mitigate KL vanishing, and a theoretical interpretation of diversity improvement. (iv) We validate the effectiveness of our model on two unconditional and one conditional generation tasks.

2 Related Work

VAE has shown great effectiveness in a wide range of NLG tasks, such as storytelling (Yu et al., 2020; Fang et al., 2021), dialogue generation (Serban et al., 2017; Bao et al., 2020) and poetry composition (Yi et al., 2021). To further improve the expressive ability of VAE, researchers propose various variants, *e.g.*, vMF-VAE (Xu and Durrett, 2018) that replaces the latent distribution with von Mises-Fisher distribution, *ml*-VAE (Bouchacourt et al., 2018) that learns multi-level latent variables, and BN-VAE (Zhu et al., 2020) that utilizes batch normalization to get a non-zero KL lower bound.

Among all variants, temporal VAE is the most prevalent one in the era of RNN, which introduces latent variables at each timestep and could natu-

rally fit with the recurrent structure of RNN. Existing temporal VAE fall into three paradigms according to the parameterization and dependence of the latent variables' prior distributions, namely *IND*, *CGD*, and *RGD*, as mentioned in Sec. 1. For example, TWR-VAE (Li et al., 2020c) utilizes a timestep-wise regularisation through independent latent variables with IND. VAD (Du et al., 2018) incorporates CGD into latent variables and augments the posterior distribution with a backward RNN. Recurrent VAE (Chung et al., 2015) learns token-wise latent variables with each sequentially conditioned on preceding ones as well as the context (*i.e.*, RGD). By modeling the trajectory of both observed text sequences and latent space, recurrent VAE could capture the sequential variability better (Goyal et al., 2017; Hajiramezanali et al., 2020). Besides, we will show that such recurrent dynamics could theoretically reinforce the dependence on the stochastic and generalized latent space, thus boosting generation diversity by a large margin.

Recently, with the flourishing of the powerful Transformer architecture, researchers have devoted to combining it with VAE for text modeling and generation (Wang and Wan, 2019; Li et al., 2020a; Fang et al., 2021; Hu et al., 2022). VAEs could promote generation diversity with satisfactory quality, benefiting from the intrinsic randomness in latent space. Therefore, VAE-based Transformers are essential for various tasks demanding creativity, such as advertising text generation (Shao et al., 2019). Two of the temporal VAE paradigms, IND and CGD, can be easily adapted into Transformer. For instance, SVT (Lin et al., 2020) applies CGD-based VAE to dialogue generation. Nonetheless, the integration of recurrent VAE is still an open challenge due to the conflict in the parallelism in Transformer and recurrent dependence of recurrent VAE. To fully exploit the expressive power of recurrence, we revisit recurrent VAE in Transformer and propose TRACE which possesses advantages of both generation diversity and training parallelism.

3 Preliminaries

3.1 VAE

As one of the representative generative models, VAE has proven to be an effective paradigm for estimating the data distribution by introducing a latent variable z and modeling the joint distribution:

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}). \tag{1}$$

---

[2] TRACE: **T**ransformer **R**ecurrent **A**utoen**C**od**E**r

The prior distribution $p(\boldsymbol{z})$ is commonly a standard Gaussian distribution. The conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ is generally parameterized by a neural network, known as the *generative network* (decoder) to recover the observed data from latent variables. Directly estimating $p(\boldsymbol{x}|\boldsymbol{z})$ brings the intractable posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. Instead, VAE introduces a variational approximation $q(\boldsymbol{z}|\boldsymbol{x})$ and derives the Evidence Lower BOund (ELBO):

$$\log p(\boldsymbol{x}) \geq \mathcal{L}_{ELBO}(\boldsymbol{x}) =$$
$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \quad (2)$$

where KL means the Kullback-Leibler divergence.

In practice, the approximated posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is parameterized as Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma^2}))$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are estimated by a neural network, known as the *inference network* (encoder). The generative network $p(\boldsymbol{x}|\boldsymbol{z})$ and inference network $q(\boldsymbol{z}|\boldsymbol{x})$ are jointly optimized by maximizing the lower bound in Eq.(2).

3.2 Temporal VAE

Unlike standard VAE, which only involves one latent variable $\boldsymbol{z}$, temporal VAE learns one latent variable at each time step. Denote $\boldsymbol{z}_t \in \mathbb{R}^l$ and $\boldsymbol{x}_t \in \mathbb{R}^h$ as the latent variables and the observed data at t-th step, respectively. Next, we will present the mathematical details of three paradigms of temporal VAE, namely IND, CGD, and RGD.

IND: The prior distribution $p(\boldsymbol{z}_t)$ follows the standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and the posterior one conditions on the preceding context as $q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq t})$. Then, we obtain the ELBO of IND:

$$\sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq t})} \log p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{x}_{<t})) \quad (3)$$
$$- \mathrm{KL}(q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq t})||p(\boldsymbol{z}_t)).$$

CGD: CGD constructs the prior distribution considering the observed text $p(\boldsymbol{z}_t|\boldsymbol{x}_{<t})$ and the posterior one based on the complete text $\boldsymbol{x} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$. The lower bound of CGD is:

$$\sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x})} \log p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{x}_{<t})) \quad (4)$$
$$- \mathrm{KL}(q(\boldsymbol{z}_t|\boldsymbol{x})||p(\boldsymbol{z}_t|\boldsymbol{x}_{<t})).$$

RGD: RGD parameterizes the generative process by the following factorization:

$$p(\boldsymbol{x}_{\leq T}, \boldsymbol{x}_{\leq T}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t}) p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t}).$$
$$(5)$$

The latent variables $\boldsymbol{z}_t$ follows the prior distribution $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})$ and the posterior one follows $q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})$. Then, we obtain the ELBO:

$$\mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \Big[\sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t})$$
$$- \mathrm{KL}(q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})||p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})) \Big], \quad (6)$$

where $q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})$ can be factorized as:

$$q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t}). \quad (7)$$

We present the detailed deduction of Eq.(6) in Appendix B.1.

In an RNN-like backbone, we can construct the representation of $\boldsymbol{x}_{\leq t}$ with the hidden states at t-th step and compute the distribution parameters of $\boldsymbol{z}_t$.

4 Method

To incorporate recurrent VAE (RGD) into Transformer, we propose TRACE that learns recurrent segment-wise latent variables and design an acceleration method to make full use of the advantage of parallelism in Transformer. We present the adaption of recurrent VAE to Transformer and residual parameterization in Sec. 4.1, demonstrate the parallel training method in Sec. 4.2, and provide a theoretical interpretation of TRACE's effectiveness for boosting diversity in Sec. 4.3.

4.1 Transformer-based Recurrent VAE

Different from the token-wise latent variables implemented in RNN-based VAEs, TRACE learns segment-wise $\boldsymbol{z}_t$ based on the representation of t-th segment, $\boldsymbol{x}_t$. We can devise different principles to separate the segments, such as the inherent separation like sentence or utterance, or specifying a fixed segment length. We add a special token [SEP] to the end of each segment.

Fig. 1 depicts the architecture of TRACE. At the encoder, we design two kinds of attention mask matrices. First, we introduce an *extra mask matrix*, a partitioned lower triangular matrix (the left of Fig. 1), which allows each token to attend to all tokens in the same segment and previous segments. Second, we design an *intra mask matrix*, a strict partitioned matrix to make each token attend to only the tokens within the same segment. We input the separated text sequence into the Transformer encoder twice, with the extra and intra mask matrix,
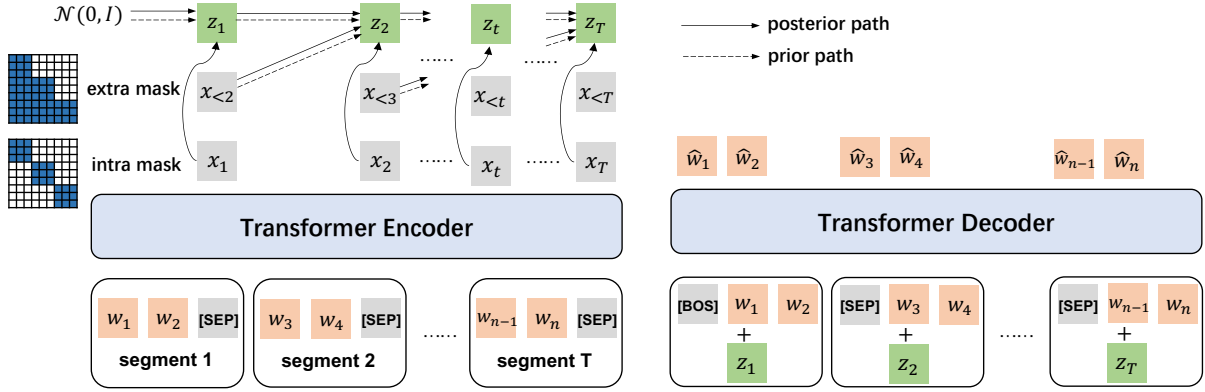
Figure 1: Architecture of TRACE. We add a special token at the end of each segment and obtain the representation of $\boldsymbol{x}_t$ and $\boldsymbol{x}_{<t}$ from the Transformer encoder (inference network) with two kinds of modified attention mask matrix. The solid and dotted lines are the posterior and prior paths, respectively. The sampled latent variables are added to the token embedding in the Transformer decoder (generator network).

respectively. Then, the output of t-th [SEP] from the final encoder layer can be used as the representation of $\boldsymbol{x}_{<t}$ and $\boldsymbol{x}_t$. Now, we can obtain the parameters of the prior distribution of $\boldsymbol{z}_t$ by:

$$p(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{<t}) = \mathcal{N}(\boldsymbol{\mu}_t, \mathrm{diag}(\boldsymbol{\sigma}_t^2)), \quad (8)$$

$$[\boldsymbol{\mu}_t, \log\boldsymbol{\sigma}_t^2] = f(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{<t}), \quad (9)$$

where f is the prior network, parameterized as linear layers $\boldsymbol{W}_{\boldsymbol{\mu}}^f, \boldsymbol{W}_{\boldsymbol{\sigma}}^f \in \mathbb{R}^{(l+h)\times l}$. The prior distribution of $\boldsymbol{z}_1$ is the standard Gaussian distribution.

For the posterior distribution, we utilize residual parameterization (Vahdat and Kautz, 2020) that parameterizes the relative difference between prior and posterior distributions. In this case, the difference lies in $\boldsymbol{x}_t$. Therefore, we can compute the posterior distribution as:

$$q(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{\leq t}) = \mathcal{N}(\boldsymbol{\mu}_t+\Delta\boldsymbol{\mu}_t, \mathrm{diag}(\boldsymbol{\sigma}_t^2\cdot\Delta\boldsymbol{\sigma}_t^2)), \quad (10)$$

$$[\Delta\boldsymbol{\mu}_t, \log\Delta\boldsymbol{\sigma}_t^2] = \mathrm{LayerNorm}(g(\boldsymbol{x}_t)), \quad (11)$$

where g is the posterior network, parameterized as $\boldsymbol{W}_{\boldsymbol{\mu}}^g, \boldsymbol{W}_{\boldsymbol{\sigma}}^g \in \mathbb{R}^{h\times l}$. We regulate the output of g by layer normalization (Ba et al., 2016). To reveal the benefits of residual parameterization and layer normalization, we give the following theorem:

Theorem 1 *The expectation of the KL term* $\mathrm{KL}(q(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{\leq t})||p(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{<t}))$ *has a lower bound* $\frac{l(\gamma^2+\beta^2)}{2\boldsymbol{\sigma}_t^2}$, *where l is the latent dimension, γ and β are the parameters of layer normalization.*

We leave the proof in Appendix B.3. Theorem 1 indicates that we can easily control the lower bound of the KL term by setting a fixed γ and hence mitigate KL vanishing. We choose layer normalization

here since it is superior to batch normalization in Transformer based models (Shen et al., 2020) (See Table 5). Besides, Theorem 1 is compatible with both unconditional and conditional generation compared to the BN VAE model (Zhu et al., 2020).

After deriving the prior and posterior distributions, we can compute the KL loss and sample the latent variables with the reparameterization trick (Kingma and Welling, 2014). The sampled latent variables $\boldsymbol{z}_t$ are injected into the Transformer decoder by adding with the input embedding.

4.2 Parallel Training for Recurrent VAE

The method above requires sequentially sampling each latent variable, which hinders the parallelism training in Transformer. For acceleration, we further design a parallel training method.

With the reparameterization trick, when sampling $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$, we actually first sample a white noise $\boldsymbol{\epsilon} \sim N(\boldsymbol{0},\boldsymbol{I})$, then get $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon}\circ\boldsymbol{\sigma}$, where $\circ$ is element-wise multiplication. We omit LayerNorm for simplicity. For each t, we have:

$$\boldsymbol{z}_t \approx \boldsymbol{v}_t + \sum_{i=1}^{t-1}\boldsymbol{u}_i, \quad (12)$$

$$\boldsymbol{v}_t = (\boldsymbol{W}_{\boldsymbol{\mu}2}^f\boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\mu}}^g\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t + \left(\boldsymbol{W}_{\boldsymbol{\sigma}2}^f\boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\sigma}}^g\boldsymbol{x}_t\right)\circ\boldsymbol{\epsilon}_t, \quad (13)$$

$$\boldsymbol{u}_t = \boldsymbol{W}_{\boldsymbol{\mu}1}^f\boldsymbol{v}_t + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f\boldsymbol{v}_t\circ\boldsymbol{\xi}_t, \quad (14)$$

where $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\xi}_i$ are independent white noises sampled from standard Gaussian distribution. $[\boldsymbol{W}_{\boldsymbol{\mu}1}^f, \boldsymbol{W}_{\boldsymbol{\mu}2}^f], [\boldsymbol{W}_{\boldsymbol{\sigma}1}^f, \boldsymbol{W}_{\boldsymbol{\sigma}2}^f]$ are the split of parameter matrices of function f with $\boldsymbol{W}_{\boldsymbol{\mu}1}^f, \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \in$

$\mathbb{R}^{l \times l}$ and $\boldsymbol{W}_{\boldsymbol{\mu}2}^{f}, \boldsymbol{W}_{\boldsymbol{\sigma}2}^{f} \in \mathbb{R}^{h \times l}$. We leave the complete derivation in Appendix B.2.

In this way, we can parallelly train the model while keeping the advantage of RGD. We can parallelly compute $\boldsymbol{v}_t$ for all time steps first, and then obtain $\boldsymbol{u}_t$ in parallel based on $\boldsymbol{v}_t$. Then we can make the matrix multiplication between the concatenation of $\boldsymbol{u}_t$ of all time steps and a lower triangular matrix of ones to parallelly obtain $\sum_{i=1}^{t-1} \boldsymbol{u}_i$. Finally, by making a sum, we get the approximation of $\boldsymbol{z}_t$. Similarly, we obtain the latent samples from the posterior distribution in Eq.(10).

In the approximation of Eq.(12), we assume that $\boldsymbol{W}_{\boldsymbol{\mu}1}^{f}$ and $\boldsymbol{W}_{\boldsymbol{\sigma}1}^{f}$ are idempotent matrices to avoid the power of matrix. However, such simplification may bring too much noise. To stabilize the training process, we adopt spectral normalization (Miyato et al., 2018) to restrain $\boldsymbol{W}_{\boldsymbol{\mu}1}^{f}$ and $\boldsymbol{W}_{\boldsymbol{\sigma}1}^{f}$.

4.3 Why Could TRACE Boost Diversity

We give a theoretical demonstration of the advantage of TRACE on the improvement of generation diversity. We have the following theorem:

Theorem 2 *Each reconstruction term in the ELBO, $\mathbb{E}_{q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t})} p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t})$, is upper bounded by $I_q(\boldsymbol{x}_t; \boldsymbol{z}_{\leq t}|\boldsymbol{x}_{<t})$, where I is the mutual information.*

The proof can be found in Appendix B.4. Based on Theorem 2, optimizing the reconstruction terms means maximizing $I(\boldsymbol{x}_t; \boldsymbol{z}_{\leq t}|\boldsymbol{x}_{<t})$, which could enhance the dependency between $\boldsymbol{x}_t$ and $\boldsymbol{z}_{\leq t}$. In this way, the model would rely more on the flexible latent space than the deterministic context text, bringing more randomness to improve diversity while maintaining satisfactory coherence.

5 Experiment

5.1 Dataset

We carry out experiments on two datasets for language modeling and unconditional generation, including the Yelp and Yahoo (Yang et al., 2017; He et al., 2019) and one dataset, WritingPrompts (WP) (Fan et al., 2018) for conditional generation. We list the detailed data statistics of these datasets in Table 1. Due to the limited computation capability, we restrain a max length of 750 for training in WP.

5.2 Implementation Details

We use the pretrained language model GPT-2 (Radford et al., 2019) as the backbone. The encoder and decoder of TRACE share the same parameters

Dataset	# Train	# Dev	# Test	Length
Yelp	100k	10k	10k	96
Yahoo	100k	10k	10k	79
WP	164k	15k	15k	421

Table 1: **Statistics of datasets**. Length means the average text length of the three datasets.

initialized with GPT-2 and are fine-tuned on the target datasets. We choose 32 for the dimension of latent space and use the cyclical annealing trick (Fu et al., 2019) during training. We set batch size as 32 and the learning rate as $5e-5$, γ in the layer normalization as 3. We separate segments with fixed length 10 on Yelp and Yahoo datasets and use the initial sentence as a segment on the WP dataset with NLTK toolkit (Bird, 2006) for segmenting. We use the top-k sampling strategy (Holtzman et al., 2020) to decode the sequence with k as 50 for all models on all datasets. When the generated token is [SEP], we sample a new latent variable based on the prior distribution to enter a new segment. We implement TRACE and other VAE baselines with open-source Huggingface Transformers (Wolf et al., 2020) library of v4.10.0 and use NVIDIA GeForce RTX 3090 to conduct all experiments.

5.3 Baseline

We compare TRACE with several transformer-based solid models. All baseline models are utilized with the same backbone model.

GPT-2: We finetune the GPT-2 model (Radford et al., 2019) on each dataset.

IND/CGD: We implement the Transformer version of the two models, TWR-VAE (Li et al., 2020c) and VAD (Du et al., 2018) which belongs to IND and CGD respectively. The segment separation is consistent with TRACE.

No Recurrence (Li et al., 2020a): We remove the recurrence and only involve one latent variable to verify the effectiveness of recurrence. The latent variables are injected into the decoder by being added with the text embedding in the decoder.

5.4 Metrics

We evaluate unconditional generation tasks with three perspectives; **(a) Representation Learning**: we report ELBO, KL, mutual information (MI) (Alemi et al., 2016) and activate units (AU) (Burda et al., 2016). We change the threshold to 0.1 when computing the AU metric to distinguish

Model	Representation Learning				Generation Quality		Generation Diversity		
	ELBO↓	KL↑	MI↑	AU↑	PPL↓	CND↓	SB↓	Dist↑	JS↓
Dataset: Yelp									
GPT-2	-	-	-	-	22.13	0.68	65.90	15.96	0.51
IND	328.17	2.32	2.92	25	15.35	0.76	61.24	16.24	0.45
CGD	327.84	0.07	0.04	2	16.29	<u>0.46</u>	68.36	13.68	0.59
No Recurrence	327.35	<u>3.89</u>	3.88	<u>26</u>	20.12	**0.43**	65.38	15.47	0.44
TRACE-R	**298.61**	3.51	**19.57**	32	**13.03**	0.47	**57.02**	**23.01**	**0.33**
TRACE-P	<u>315.28</u>	**4.52**	<u>5.25</u>	32	<u>14.88</u>	0.82	<u>60.25</u>	16.58	<u>0.40</u>
Dataset: Yahoo									
GPT-2	-	-	-	-	24.17	0.55	54.06	<u>21.07</u>	<u>0.28</u>
IND	285.61	2.31	2.89	20	17.05	0.90	53.08	20.55	0.45
CGD	285.92	0.17	0.12	3	18.84	**0.45**	58.81	18.12	0.53
No Recurrence	286.89	3.62	5.65	<u>25</u>	21.18	**0.45**	54.15	20.78	0.32
TRACE-R	**258.25**	<u>3.65</u>	**43.20**	32	**14.56**	<u>0.53</u>	**47.33**	**26.35**	**0.26**
TRACE-P	<u>278.35</u>	**3.99**	<u>8.72</u>	32	<u>16.85</u>	0.78	<u>52.75</u>	20.61	0.30

Table 2: Evaluation results for unconditional generation. SB: Self-BLEU. TRACE-R: TRACE with standard RGD. TRACE-P: The parallel version of TRACE. The best/second best results are in **bold** and <u>underlined</u>, respectively.

different models further. **(b) Generation Quality**: we report PPL and CND (Li et al., 2020b) to evaluate the generation capacity of models. Different from standard auto-regressive language models like GPT-2, VAE-based models could not estimate exact PPL. Therefore, following He et al. (2019), we use importance-weighted samples to approximate $\log p(\boldsymbol{x})$ and estimate PPL. CND measures the divergence between the generated one and the ground truth testset. **(c) Generation Diversity**: we report Self-BLEU (Zhu et al., 2018), Dist (Li et al., 2016) and JS (Jaccard similarity) (Wang and Wan, 2018) to evaluate the diversity of generated text.

We consider the quality and diversity in story generation. We report BLEU (Papineni et al., 2002), Rouge-1, 2, L (Lin and Hovy, 2002), and BERTScore (Zhang et al., 2020) to evaluate the quality of generated samples, and the same diversity metrics used in unconditional generation. More details about metrics are listed in Appendix A.1.

5.5 Results

5.5.1 Unconditional Generation

We present the results of the unconditional generation task on Yelp and Yahoo datasets in Table 2. As the results show, TRACE achieves significant improvement on most metrics. Better ELBO and MI indicate TRACE has stronger capability of representation learning. Especially, considerably higher MI empirically validates Theorem 2, which

manifests that with the RGD mechanism, the observed data will be connected to the latent space more closely. Higher KL and AU also empirically show the benefit of residual parameterization. Besides, lower PPL and comparable CND indicates acceptable quality of the text generated by TRACE.

Moreover, TRACE can produce much more diverse text compared with baselines. Among the baselines, the CGD baseline suffers from the KL vanishing problem, which means the decoder ignores the latent variables. Therefore, without the randomness arising from latent variables, CGD performs the worst on generation diversity. In contrast, TRACE obtains the most prominent enhancement on all diversity metrics. Such improvement originates from two aspects: First, compared to GPT-2 and No Recurrence, sampling latent variables at each time step brings extra randomness for the output. Second, Unlike the other two temporal VAE baselines, TRACE is endowed with the theoretical advantage of RGD, which strengthens the interaction between the text and latent space and then absorbs much flexibility from the generalized latent space. IND simply increases randomness from standard Gaussian prior distribution, which negatively affects quality while causing limited diversity.

Lastly, comparing TRACE-R and TRACE-P, despite the slight drop of quality and diversity, TRACE-P still outperforms baselines on most metrics except CND. Such marginal cost is accept-

Model	Quality					Diversity		
	BLEU↑	Rouge-1↑	Rouge-2↑	Rouge-L↑	BertScore↑	SB↓	Dist↑	JS↓
GPT-2	27.89	27.72	3.92	10.18	78.12	53.78	22.99	0.51
IND	31.17	32.44	4.35	11.39	**81.31**	67.44	13.69	1.34
CGD	**31.65**	**32.56**	**4.46**	**11.59**	81.28	68.25	14.87	1.97
No Recurrence	31.57	32.27	4.28	11.30	81.25	67.25	13.51	1.38
TRACE-R	30.65	31.80	4.12	11.15	81.14	64.11	13.18	0.84
TRACE-P	28.55	29.52	3.48	10.98	80.63	**47.26**	**26.17**	**0.70**

Table 3: Evaluation results for conditional generation.

able considering the acceleration with parallelism. See the comparison of training speeds in Sec. 5.8. The performance loss is mainly caused by the last two approximation steps in Appendix B.2. Theoretically, we approximate the multiplication of t independent random variables as one Gaussian distribution using the central limit theorem (holds for infinite length) in Eq.(35), and restrict two parameter matrices to be idempotent (holds when the eigenvalue is 1 or 0) in Eq.(36). In practice, these assumptions don't hold because the sequence length is not infinite, and the spectral normalization only restrains the largest eigenvalue (strictly limiting makes optimization difficult). Therefore, empirically we hurt the interaction of latent variables and sacrifice some information in them, leading to decreased performance compared to TRACE-R.

5.5.2 Conditional Generation

We report the results of conditional generation on the WP dataset. As shown in Table 3, TRACE achieves comparable generation performance on quality (still better than GPT-2) and significant improvement on diversity (especially on Self-Bleu and Dist compared with other VAE baselines). Although the quality of text generated by TRACE-P is relatively defective, TRACE-P still outperforms GPT-2 on both quality and diversity. The overall enhancement of diversity empirically further validates our theoretical analysis.

Interestingly, TRACE-P achieves better generation diversity than TRACE-R on WP, which is opposite to the results on both Yelp and Yahoo datasets. This contrary tendency of diversity mainly originates from different sequence lengths. On Yelp and Yahoo datasets, whose text are relatively short, TRACE-R performs better. However, on the WP dataset with much longer text, the approximation in TRACE-P reduces the exploitation of $x_{<t}$ by reducing the interaction of each x_t, as shown in

Model	Fluency	Coherence	Novelty
GPT2	2.18	2.28	<u>2.46</u>
IND	2.38	2.41	2.40
CGD	**2.43**	<u>2.47</u>	2.41
No Recurrence	<u>2.42</u>	**2.51**	2.44
TRACE-R	2.40	2.46	**2.55**
TRACE-P	2.24	2.32	<u>2.46</u>

Table 4: Human evaluation results on the WP dataset. The scores range from 1 (worst) to 5 (best). The p-value < 0.01, and the Kappa score is 0.61, which means the evaluation is in an acceptable inter-annotator agreement.

Eq.(36), which loosens the dependency on context and forces TRACE-P to produce more uncertain text (better diversity but lower quality) than TRACE-R.

5.6 Human Evaluation

We conduct the human evaluation on the WP dataset. We generate 50 samples given the source input from the testset with each model and invite five proficient annotators to access the generated text by scoring three criteria: **Fluency** (whether generated text are syntactically fluent), **Coherence** (whether the generated part is consistently structured and coherent with input), and **Novelty** (whether each generated instance is novel and distinct), which cover both generation quality and quality we care about. See Appendix A.2 for more evaluation details.

We report the evaluation results in Table 4. TRACE obtains satisfactory results on Fluency and Coherence but stands superior to other baselines on Novelty. The results of the human evaluation are consistent with automatic evaluations.

5.7 Ablation Study

Table 5 shows the results of the ablation study on the Yelp dataset. We mainly justify the effective-
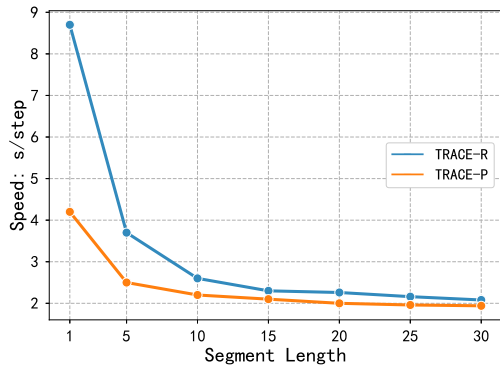
Figure 2: Training speed (average seconds per step) of TRACE-R and TRACE-P with different segment lengths.
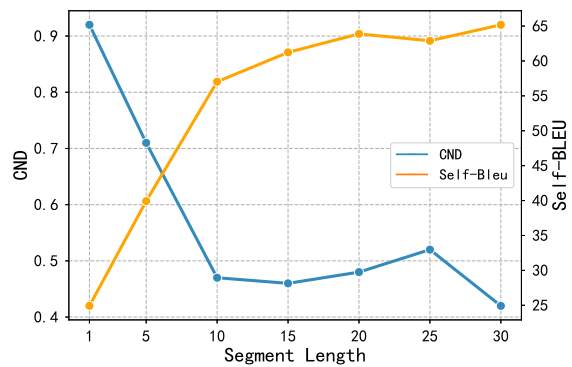

Figure 3: CND and Self-Bleu achieved by TRACE-R with varying segment length.

Model	ELBO↓	CND↓	SB↓	Dist↑	JS↓
TRACE-R	298.61	0.47	57.02	23.01	0.33
-LN	307.33	0.49	64.89	15.37	0.43
-RP	326.88	0.78	67.89	13.87	0.58
+BN	305.02	0.63	59.25	22.88	0.35
TRACE-P	315.28	0.82	60.25	16.58	0.40
-SN	318.25	0.91	62.32	17.22	0.43

Table 5: Ablation study on Yelp. +BN means replacing layer normalization with batch normalization. -LN, -RP and -SN means removing layer normalization, residual design and spectral normalization, respectively.

Model	CND↓	Self-Bleu↓	Dist↑
Greedy decoding			
IND	7.88	93.81	4.31
CGD	55.27	99.56	0.87
No Recurrence	4.06	94.01	4.36
TRACE-R	**2.01**	**78.11**	**9.28**
Beam search, beam size=10			
IND	76.27	97.18	0.89
CGD	266.52	99.82	0.08
No Recurrence	63.12	97.41	0.85
TRACE-R	**36.28**	**87.56**	**6.09**
Top-k, k=50			
IND	0.76	61.24	16.24
CGD	0.46	68.36	13.68
No Recurrence	**0.43**	65.38	15.47
TRACE-R	0.47	**57.02**	**23.01**

Table 6: Comparison of different decoding strategies on Yelp Dataset.

ness of residual parameterization, layer normalization used after the posterior network, and spectral normalization for the prior network in TRACE-P. We also compare batch normalization by replacing layer normalization with batch normalization. Experimental results show that all of them benefit the quality or diversity. Specifically, without layer normalization or residual parameterization, TRACE will tend to ignore the latent variables and loss the diversity brought by the recurrent structure during generation. The difference between batch normalization and layer normalization is relatively marginal, while the former still performs better.

5.8 Analysis

Training Speed We compare the training speed of TRACE-R and TRACE-P on Yelp dataset with different pre-defined fixed segment lengths. Small segment length will lead to more recurrence steps in TRACE-R. As shown in Fig. 2, small segment length leads to increased number of segments, resulting in much more training time for TRACE-R. In contrast, for TRACE-P, our proposed parallel training method remarkably shortens the train-

ing time. When segment length is 1 (token-wise), the training of TRACE-P is more than twice faster than that of TRACE-R. When segment length is 5, TRACE-P is still 50% faster than TRACE-R. As the segment length grows, the number of segments decreases, and consequently, the training speeds of TRACE-R and TRACE-P reach unanimity. Such empirical results generally confirm the effectiveness of TRACE-P's parallelism. Besides, it is worth emphasizing that despite the deceleration of TRACE-R in the training phase, the inference process is hardly influenced by the recurrence structure because of the auto-regressive decoding manner. Since the efficiency bottleneck of deploying NLG models mainly lies in inference, we believe the speed of TRACE-R is still practical enough for downstream NLG tasks.

Prompt: When everyone turns 18 , they receive a pet which is figurative of their personality. You 're the first person to receive a **dragon**...

IND: When people turn eighteen, they get an animal which represents them as one and can be found everywhere in their town. We didn't really ask for it though - only that we should at least know how it worked. Nowadays we had to choose between the puppy dog and cat......

TRACE-R: I have been waiting for this moment ever since the **wizards** gave me my **special ability** at elementary school. They said I was one step ahead of everyone else, and that your strength can **grow exponentially** if you use your abilities **dragon** gives you wisely and carefully......

Figure 4: Samples generated by different models on the WP dataset. TRACE-R produces responses to the dragon in the prompt and imagines an engaging story that the protagonist was given a special ability by wizards and would begin an adventure with his pet dragon.

Separation of Segment   We explore the influence of segment length on the performance of TRACE-R by evaluating CND for generation quality and Self-Bleu for diversity. As Fig. 3 shows, the generation diversity drops with the increase of segment length. It is consistent with intuition that longer segment leads to looser context correlations but enhanced sampling randomness and thus improved diversity.

Decoding Strategy   We evaluate the performance of TRACE-R on generation quality and diversity with different decoding strategies, including greedy decoding and beam search with beam size $= 10$. In these cases, the randomness only originates from sampling the latent variables. As shown in Table 6, by recurrently sampling latent variables which interact with hidden states, TRACE has more distinct advantages in generating diverse text with greedy decoding or beam search. In contrast, sampling decoding itself can enhance generation randomness and thus dilute the diversity improvement obtained by our model. Therefore, using sampling decoding becomes the most difficult case for further improving diversity. We select such a challenging setting in main experiments to verify the effectiveness of TRACE and find it still achieves non-trivial enhancement on diversity, demonstrating the robustness of TRACE in various decoding methods.

5.9   Case Study

Fig. 4 gives one generation example of TRACE-R and IND from WP dataset. The input prompt mentioned that people receive a dragon as a pet. The generated text of IND only talks about receiving an animal. In contrast, the generation of TRACE-R first tells that wizards gave him the ability, and mentions the ability could grow with the dragon. We can see the text produced by our model tells an engaging story like a warrior would go to fight with his given dragon. In general, TRACE-R generates a more vivid continuation compared with IND.

6   Conclusion

In this paper, we revisit the recurrent VAE framework prevalent in the era of RNN, and propose a novel Transformer-based recurrent VAE structure TRACE. TRACE learns a series of segment-wise latent variables conditioned on the preceding ones. We establish the latent distributions with novel residual parameterization. To accelerate training, we design an approximate algorithm of learning latent variables to fit with the Transformer framework. Experimental shows that TRACE achieves significant improvement in generation diversity based on the tight relationship between text hidden states and the latent space. In the future, we will further explore the potential of TRACE in other larger pretrained models like GPT-3.

Acknowledgement

Limitations

While TRACE achieves significant improvement in generation diversity, it still has some limitations. First, the trade-off between quality and diversity is a common problem in natural language generation. TRACE is not an exception. The diversity of TRACE's increases while the quality inevitably drops a little. We will further explore better methods to balance quality and diversity. Second, our parallel acceleration training method requires certain approximations, which, to some extent, hurts the initial advantage of recurrence and leads to a drop in the quality. We will continue to design better acceleration training methods. Third, the speedup of our parallel version of TRACE is limited. Under the practical segment length setting (*e.g.*, 10 and 20), the acceleration is marginal. We also plan to further promote our methods to benefit faster training in the future.

References

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

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv: Machine Learning*.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96, Online. Association for Computational Linguistics.

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, Sydney, Australia. Association for Computational Linguistics.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. 2018. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

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

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. 2018. Variational autoregressive decoder for neural response generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3154–3163, Brussels, Belgium. Association for Computational Linguistics.

Otto Fabius, Joost R van Amersfoort, and Diederik P Kingma. 2015. Variational recurrent auto-encoders. In *ICLR (Workshop)*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. 2021. Transformer-based conditional variational autoencoder for controllable story generation. *arXiv: Computation and Language*.

Jean-Yves Franceschi, Edouard Delasalles, Mickael Chen, Sylvain Lamprier, and Patrick Gallinari. 2020. Stochastic latent residual video prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3233–3246. PMLR.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250, Minneapolis, Minnesota. Association for Computational Linguistics.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, volume 30.

Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna Narayanan, Mingyuan Zhou, and Xiaoning Qian. 2020. Semi-implicit stochastic recurrent neural networks. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3342–3346. IEEE.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Jinyi Hu, Xiaoyuan Yi, Wenhao Li, Maosong Sun, and Xing Xie. 2022. Fuse it more deeply! a variational transformer with layer-wise latent variable inference for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 697–716, Seattle, United States. Association for Computational Linguistics.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential latent knowledge selection for knowledge-grounded dialogue. *International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020a. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.

Jianing Li, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2020b. On the relation between quality-diversity evaluation and distribution-fitting goal in text generation. In *International Conference on Machine Learning*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Longyuan Li, Junchi Yan, Xiaokang Yang, and Yaohui Jin. 2019. Learning interpretable deep state space model for probabilistic time series forecasting. *International Joint conference on Artificial Intelligence*.

Ruizhe Li, Xiao Li, Guanyi Chen, and Chenghua Lin. 2020c. Improving variational autoencoder for text modelling with timestep-wise regularisation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2381–2397, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 45–51, Phildadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Zhaojiang Lin, Genta Indra Winata, Peng Xu, Zihan Liu, and Pascale Fung. 2020. Variational transformers for diverse response generation. *arXiv preprint arXiv:2003.12738*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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. PMLR.

Iulian 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Harshil Shah and David Barber. 2018. Generative neural machine translation. *Advances in Neural Information Processing Systems*, 31.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3257–3268, Hong Kong, China. Association for Computational Linguistics.

Sheng Shen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. 2020. Powernorm: Rethinking batch normalization in transformers. In *International Conference on Machine Learning*, pages 8741–8751. PMLR.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder variational autoencoders. In *Neural Information Processing Systems*.

Arash Vahdat and Jan Kautz. 2020. Nvae: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems*.

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.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *International Joint Conference on Artificial Intelligence*.

Tianming Wang and Xiaojun Wan. 2019. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *IJCAI*, pages 5233–5239.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513, Brussels, Belgium. Association for Computational Linguistics.

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. PMLR.

Xiaoyuan Yi, Ruoyu Li, Cheng Yang, Wenhao Li, and Maosong Sun. 2020. Mixpoet: Diverse poetry generation via learning controllable mixed latent space. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York, USA.

Xiaoyuan Yi, Zhenghao Liu, Wenhao Li, and Maosong Sun. 2021. Text style transfer via learning style instance supported latent space. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3801–3807.

Meng-Hsuan Yu, Juntao Li, Danyang Liu, Dongyan Zhao, Rui Yan, Bo Tang, and Haisong Zhang. 2020. Draft and edit: Automatic storytelling through multipass hierarchical conditional variational autoencoder. In *National Conference on Artificial Intelligence*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2636–2649, Online. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

A Experiment Details

A.1 Metrics Details

Perplexity (PPL). In the VAE-based framework, we commonly use importance weighted sampling to estimate the PPL. Sampling n latent variables $z_1, z_2, \ldots, z_n$ from the variational posterior distribution $q(\boldsymbol{z}_i|\boldsymbol{x})$, we have:

$$\mathcal{L}_k = \mathbb{E}\left[\log \frac{1}{n}\sum_{i=1}^{n}\frac{p(\boldsymbol{x},\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x})}\right] \leq \log \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{p(\boldsymbol{x},\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x})}\right] = \log p(\boldsymbol{x}). \tag{15}$$

As illustrated in (Burda et al., 2016), when $n \to \infty$, $\mathcal{L}_n \to \log p(\boldsymbol{x})$. Therefore, we use $\mathcal{L}_n$ to estimate $\log p(\boldsymbol{x})$ and calculate PPL with $n = 100$.

Mutual Information(MI) (Alemi et al., 2016). Mutual Information $\mathcal{I}(\boldsymbol{x}, \boldsymbol{z})$ measures the mutual dependence between $\boldsymbol{x}$ and $\boldsymbol{z}$, which is defined as:

$$\mathcal{I}_q(\boldsymbol{x}, \boldsymbol{z}) = \mathbb{E}_{p(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} \log q(\boldsymbol{z}|\boldsymbol{x}) - \mathbb{E}_{q(\boldsymbol{z})} \log q(\boldsymbol{z}), \tag{16}$$

Here, $q_{(\boldsymbol{z})} = \mathbb{E}_{p(\boldsymbol{x})}q(\boldsymbol{z}|\boldsymbol{x})$ is the aggregated posterior.

Activate Units(AU) (Burda et al., 2016). AU is computed as $A_{\boldsymbol{z}} = \mathrm{Cov}_{\boldsymbol{x}}(\mathbb{E}_{z\sim q(z|\boldsymbol{x})}[z]) > \delta$, where δ is a threshold. We use AU to measure the active units in latent variables. To more clearly distinguish the performance of TRACE and other baseline models, we increase the threshold to 0.1 from the commonly used 0.01.

CND (Li et al., 2020b). CND measures the similarity between the generated samples and the reference testset by approximating the divergence between these two distributions in n-gram spaces.

BLEU (Papineni et al., 2002). BLEU evaluates the similarity between the generated text and ground truth by measuring the n-gram overlap of generated samples and references.

Rouge (Lin and Hovy, 2002). Rouge also calculates the proportion of n-gram overlap between generated examples and reference samples, commonly used in evaluating text summarization.

BERTScore (Zhang et al., 2020). BERTScore computes the cosine similarity of representations of generated and reference text obtained by pre-trained BERT (Devlin et al., 2019).

Self-BLEU (Zhu et al., 2018). Self-Bleu computes the BLEU score among the generated text by averaging the BLEU score of each instance with all other samples as references. Lower Self-BLEU means the smaller overlap and higher diversity within the generated samples.

Dist (Li et al., 2016). Dist computes the ratio of different n-grams among the generated text.

Jaccard Similarity(JS) (Wang and Wan, 2018). JS computes the Jaccard similarity between every two generated samples.

A.2 Human Evaluation Details

We select 50 prompts from the WP dataset as input to the TRACE and other four baseline models to generate the continuations. We invite five annotators proficient in English to score the generated samples ranging from 1-5 with three criteria: Fluency, Coherence, and Novelty. During the evaluation, each annotator is given 20 groups of samples. Each group has six samples generated by six models, and the order of samples in each group is shuffled to avoid bias. Each sample is scored by two annotators and we average the evaluating score of all samples for each model, reported in the Table 4.

B Additional Proof

B.1 Deduction of ELBO of RGD

The lower bound for standard VAE is:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})} \tag{17}$$

Rewriting the $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x}, \boldsymbol{z})$ with RGD's factorization, we obtain:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \log \frac{\prod\limits_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t}) p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})}{\prod\limits_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})} \tag{18}$$

$$= \mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t}) + \log p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t}) - \log q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t}) \tag{19}$$

$$= \mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t}) - \log \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})}{p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})} \tag{20}$$

The first term in Eq.(20) is the construction term of ELBO. For the second term, we have:

$$\mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \sum_{t=1}^{T} \log \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})}{p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})} \tag{21}$$

$$= \int \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t}) \sum_{t=1}^{T} \log \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})}{p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})} \, \mathrm{d}\boldsymbol{z}_1 \ldots \boldsymbol{z}_T \tag{22}$$

$$= \mathbb{E}_{q(\boldsymbol{z}_{\leq T-1}|\boldsymbol{x}_{\leq T-1})} \mathrm{KL}\Big(q(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x}_{\leq T})||p(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x}_{<T})\Big) + \tag{23}$$

$$\mathbb{E}_{q(\boldsymbol{z}_{\leq T-2}|\boldsymbol{x}_{\leq T-2})} \mathrm{KL}\Big(q(\boldsymbol{z}_{T-1}|\boldsymbol{z}_{<T-1}, \boldsymbol{x}_{\leq T-1})||p(\boldsymbol{z}_{T-1}|\boldsymbol{z}_{<T-1}, \boldsymbol{x}_{<T-1})\Big) + \tag{24}$$

$$\cdots + \mathrm{KL}(q(\boldsymbol{z}_1|\boldsymbol{x}_1)||p(\boldsymbol{z}_1)) \tag{25}$$

$$= \mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})} \sum_{t=1}^{T} \mathrm{KL}\Big(q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{\leq t})||p(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})\Big) \tag{26}$$

B.2 Deduction of Parallel Training

$$\boldsymbol{z}_t = f_{\boldsymbol{\mu}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{<t}) + g_{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \exp\Big(f_{\boldsymbol{\sigma}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{<t}) + g_{\boldsymbol{\sigma}}(\boldsymbol{x}_t)\Big) \circ \boldsymbol{\epsilon}_t \tag{27}$$

$$\approx f_{\boldsymbol{\mu}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{<t}) + g_{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \Big(f_{\boldsymbol{\sigma}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_{<t}) + g_{\boldsymbol{\sigma}}(\boldsymbol{x}_t) + 1\Big) \circ \boldsymbol{\epsilon}_t \tag{28}$$

$$= \boldsymbol{W}_{\boldsymbol{\mu}1}^f \boldsymbol{z}_{t-1} + \boldsymbol{W}_{\boldsymbol{\mu}2}^f \boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\mu}}^g \boldsymbol{x}_t + \Big(\boldsymbol{W}_{\boldsymbol{\sigma}1}^f \boldsymbol{z}_{t-1} + \boldsymbol{W}_{\boldsymbol{\sigma}2}^f \boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\sigma}}^g \boldsymbol{x}_t\Big) \circ \boldsymbol{\epsilon}_t + \boldsymbol{\epsilon}_t \tag{29}$$

$$= \Big[\boldsymbol{W}_{\boldsymbol{\mu}2}^f \boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\mu}}^g \boldsymbol{x}_t + \Big(\boldsymbol{W}_{\boldsymbol{\sigma}2}^f \boldsymbol{x}_{<t} + \boldsymbol{W}_{\boldsymbol{\sigma}}^g \boldsymbol{x}_t\Big) \circ \boldsymbol{\epsilon}_t + \boldsymbol{\epsilon}_t\Big] + \Big[\boldsymbol{W}_{\boldsymbol{\mu}1}^f \boldsymbol{z}_{t-1} + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \boldsymbol{z}_{t-1} \circ \boldsymbol{\epsilon}_t\Big] \tag{30}$$

$$= \boldsymbol{v}_t + (\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_t) \boldsymbol{z}_{t-1} \tag{31}$$

$$= \boldsymbol{v}_t + \Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_t\Big)\Big(\boldsymbol{v}_{t-1} + (\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_{t-1}) \boldsymbol{z}_{t-2}\Big) \tag{32}$$

$$= \boldsymbol{v}_t + \Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_t\Big) \boldsymbol{v}_{t-1} + \Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_t\Big)\Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_{t-1}\Big) \boldsymbol{z}_{t-2} \tag{33}$$

$$= \boldsymbol{v}_t + \sum_{i=1}^{t-1} \prod_{j=i+1}^{t} \Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\epsilon}_j\Big) \boldsymbol{v}_i \tag{34}$$

$$\approx \boldsymbol{v}_t + \sum_{i=1}^{t-1} \Big((\boldsymbol{W}_{\boldsymbol{\mu}1}^f)^n + (\boldsymbol{W}_{\boldsymbol{\sigma}1}^f)^n \circ \boldsymbol{\xi}_j\Big) \boldsymbol{v}_i \tag{35}$$

$$\approx \boldsymbol{v}_t + \sum_{i=1}^{t-1} \Big(\boldsymbol{W}_{\boldsymbol{\mu}1}^f + \boldsymbol{W}_{\boldsymbol{\sigma}1}^f \circ \boldsymbol{\xi}_j\Big) \boldsymbol{v}_i = \boldsymbol{v}_t + \sum_{i=1}^{t-1} \boldsymbol{u}_i \tag{36}$$

In Eq.(28), we use the approximate equality $\exp x \approx x + 1$ when x is close to 0. In Eq.(35), we use one random variable to approximate the multiplication of t independent random variables based on central limit theorem. In Eq.(36), we approximate the matrix $\boldsymbol{W}_{\boldsymbol{\mu}1}^f$ and $\boldsymbol{W}_{\boldsymbol{\sigma}1}^f$ as idempotent matrix.

B.3 Proof of Theorem 1

We expand the KL term as:

$$
\begin{aligned}
&\mathrm{KL}(q(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{\leq t})||p(\boldsymbol{z}_t|\boldsymbol{z}_{<t},\boldsymbol{x}_{<t})) \\
&= \frac{1}{2}(\frac{\Delta\boldsymbol{\mu}_t^2}{\boldsymbol{\sigma}_t^2} + \Delta\boldsymbol{\sigma}_t^2 - \log\Delta\boldsymbol{\sigma}_t^2 - 1)
\end{aligned}
\tag{37}
$$

It is easy to find that $\Delta\boldsymbol{\sigma}_t^2 - \log\Delta\boldsymbol{\sigma}_t^2 - 1 \geq 0$ holds true. So we can control the lower bound of the KL term with $\Delta\boldsymbol{\mu}_t^2$. Following the application of batch normalization to VAE (Zhu et al., 2020), we regulate layer normalization (Ba et al., 2016) to the output of the posterior network, so we have:

$$
\mathbb{E}(\Delta\boldsymbol{\mu}_t^2) \geq \frac{l(\gamma^2 + \beta^2)}{2},
\tag{38}
$$

where γ and β are the parameters in layer normalization. Therefore, the KL term has a lower bound:

$$
\frac{l(\gamma^2 + \beta^2)}{2\boldsymbol{\sigma}_t^2}.
\tag{39}
$$

B.4 Proof of Theorem 2

We expand the target upper bound as follow:

$$
I_q(\boldsymbol{x}_t;\boldsymbol{z}_t|\boldsymbol{x}_{<t}) = \iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})\log\frac{q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})q(\boldsymbol{z}_t|\boldsymbol{x}_{<t})}\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{40}
$$

$$
= \iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})\log\frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})}\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{41}
$$

$$
= \iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})\log\frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)}{p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)}\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}+
\tag{42}
$$

$$
\iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})\log\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})}\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{43}
$$

$$
= \mathbb{E}_{q(\boldsymbol{z}_t,\boldsymbol{x}_{<t})}\mathrm{KL}(q(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)||p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t))+
\tag{44}
$$

$$
\iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{x}_{<t})\log\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})}\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{45}
$$

$$
\geq \iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{z}_t|\boldsymbol{x}_t,\boldsymbol{x}_{<t})q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})\log p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}-
\tag{46}
$$

$$
\iiint q(\boldsymbol{x}_{<t})q(\boldsymbol{z}_t|\boldsymbol{x}_t,\boldsymbol{x}_{<t})q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})\log q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{47}
$$

$$
\geq \iiint q(\boldsymbol{x}_{<t})p(\boldsymbol{z}_t|\boldsymbol{x}_t,\boldsymbol{x}_{<t})q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})\log p(\boldsymbol{x}_t|\boldsymbol{x}_{<t},\boldsymbol{z}_t)\,\mathrm{d}\boldsymbol{x}_t\,\mathrm{d}\boldsymbol{z}_t\,\mathrm{d}\boldsymbol{x}_{<t}
\tag{48}
$$

$$
= \mathbb{E}_{q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t})}\log p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t},\boldsymbol{z}_t).
\tag{49}
$$

In Eq.(46), the inequality holds true because the KL value is always great than 0. In Eq.(48), the inequality holds true because $-\log q(\boldsymbol{x}_t|\boldsymbol{x}_{<t})$ is always greater than 0.