Fuse It More Deeply! A Variational Transformer with Layer-Wise Latent Variable Inference for 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, China
[2] Beijing National Research Center for Information Science and Technology
[3] Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[4] International Innovation Center of Tsinghua University, Shanghai, China
[5] Microsoft Research Asia
hu-jy21@mails.tsinghua.edu.cn, xiaoyuanyi@microsoft.com

Abstract

The past several years have witnessed Variational Auto-Encoder's superiority in various text generation tasks. However, due to the sequential nature of the text, auto-regressive decoders tend to ignore latent variables and then reduce to simple language models, known as the *KL vanishing* problem, which would further deteriorate when VAE is combined with Transformer-based structures. To ameliorate this problem, we propose DELLA, a novel variational Transformer framework. DELLA learns a series of layer-wise latent variables with each inferred from those of lower layers and tightly coupled with the hidden states by low-rank tensor product. In this way, DELLA forces these posterior latent variables to be fused deeply with the whole computation path and hence incorporate more information. We theoretically demonstrate that our method can be regarded as entangling latent variables to avoid posterior information decrease through layers, enabling DELLA to get higher non-zero KL values even without any annealing or thresholding tricks. Experiments on four unconditional and three conditional generation tasks show that DELLA could better alleviate KL vanishing and improve both quality and diversity compared to several strong baselines.

1 Introduction

Variational Autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) has proven to be successful in generating various kinds of text, such as stylistic text (Hu et al., 2017; John et al., 2019), dialogue (Zhao et al., 2017), story (Yu et al., 2020) and poetry (Yi et al., 2020). The sequential nature of the text leads to typically used auto-regressive decoders in VAE for language generation. However, such strong decoders tend to evade the difficulty of learning meaningful latent codes by heavily relying on previously generated words

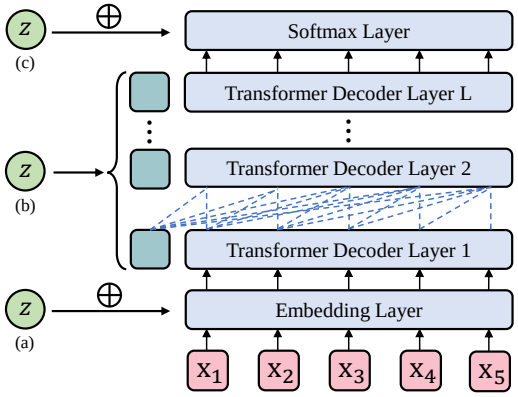

Figure 1: Existing paradigms of Transformer VAE.

and hence ignore latent variables (Bowman et al., 2016), known as *KL vanishing* or *posterior collapse*. This problem causes two drawbacks: (a) the posterior distribution quickly turns into the prior one (usually standard Gaussian), falling to build expressive latent representations; (b) the decoder reduces to a naive language model, resulting in monotonous generated text (Fu et al., 2019).

To ameliorate this problem, researchers have designed various techniques. Among them, three broadly used methods include weakening decoders (Bowman et al., 2016; Semeniuta et al., 2017; Zhao et al., 2017), KL annealing (Bowman et al., 2016; Fu et al., 2019) and KL threshold (Kingma et al., 2016; Higgins et al., 2017; Li et al., 2019). Nonetheless, the weakening of decoders restrains models' language modelling capability; annealing hyperparameters are hard to tune; KL threshold introduces a non-smooth objective with some optimization difficulties.

In the era of RNN, VAE can be easily incorporated by using the latent variable as the initial decoder state, while how to combine VAE with recently prevalent Transformer (Vaswani et al., 2017) architectures, which have made a breakthrough in text generation, still remains an open challenge.

*Corresponding author. Email: sms@tsinghua.edu.cn

As shown in Fig.1, existing methods of integrating Transformer into VAE fall into three main paradigms: (a) directly adding latent variables to input token embeddings (abbr. *Embedding*) (Li et al., 2020a); (b) using latent variables as a separate memory token vector to be attended by self-attention in each layer (abbr. *Memory*) (Fang et al., 2021); (c) combining latent variables with the last-layer decoder states before output softmax (abbr. *Softmax*) (Wang and Wan, 2019). However, paradigm (a) brings noise for self-attention. In paradigm (b), memory vectors tend to be ignored by attention, even exacerbating KL vanishing. In paradigm (c), latent variables couldn't deeply interfere with the whole computation path. Sec.3.3 presents more detailed analyses.

To better incorporate Transformer into VAE and theoretically ameliorate the *KL vanishing* problem, we propose DELLA[1], a novel variational transformer framework. DELLA learns a series of layer-wise latent variables in a Transformer encoder, and each is inferred from those of lower layers and then tightly coupled with the hidden states in the corresponding decoder layer by low-rank tensor product. Our method theoretically stimulates the entanglement of latent variables and hence allows propagation of undiminished latent information through layers. As a result, DELLA forces posterior latent variables to be deeply fused with the entire computation path and encode richer information of input text, achieving higher KL values even without any annealing or threshold training tricks.

In summary, our contributions are as follow: (i) We are the first to propose layer-wise inferred latent variables in Transformer-based architecture to mitigate KL vanishing; We (ii) innovatively inject latent variables using low-rank tensor product, (iii) provide a theoretical validity of our method and (iv) demonstrate its effectiveness on four unconditional and three conditional generation tasks. Our codes are available at https://github.com/OpenVLG/DELLA.git.

2 Related Work

Thanks to the representation capacity of latent space, VAE has been widely adopted for both image generation (van den Oord et al., 2017; Vahdat and Kautz, 2020) and text generation (Bowman et al., 2016; Hu et al., 2017). In the early stage, VAE was combined with RNN decoders for generating a broad range of text, varying from dialogue (Serban et al., 2016), image caption (Wang et al., 2017), text summarization (Gupta et al., 2017) to story (Yu et al., 2020) and poetry (Yi et al., 2020). In this case, latent variables are usually utilized as either the initial decoder state (Li et al., 2018) or input at each time step (Gupta et al., 2017).

In spite of extensive applications, VAE suffered from *KL vanishing* in the scenario of text generation (Bowman et al., 2016). Several lines of techniques have been proposed to alleviate this problem. The first line is to avoid a too fast decrease of the KL divergence by re-weighting. KL annealing (Bowman et al., 2016) linearly increased the weight of KL term from 0 to 1 during the warm-up period. Fu et al. (2019) further proposed cyclical annealing, which repeats the warm-up process multiple times. The second line guarantees a positive lower bound of the KL term. KL thresholding (Kingma et al., 2016) achieved a fixed minimum by combining a hinge loss, while BN-VAE (Zhu et al., 2020) learned more flexible ones via batch normalization. δ-VAE (Razavi et al., 2019) chose to restrain the family of posterior distributions. The third line aims to constraint decoders to force a more informative latent variable. Wang et al. (2017) introduced an auxiliary BOW (bag-of-words) loss. He et al. (2019) added additional training loops for the encoder. Yang et al. (2017) adopted dilated CNN as decoder, and Dieng et al. (2019) added skip connections to the decoder. Although the above methods mitigate *KL vanishing* to some extent, it is still challenging for either tuning or optimization.

In these years, the powerful Transformer has been integrated with VAE to benefit diverse tasks, including text classification (Gururangan et al., 2019), story generation (Wang and Wan, 2019; Fang et al., 2021) and dialogue generation (Lin et al., 2020). Optimus (Li et al., 2020a) further bridged the pre-trained BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) with VAE for pre-training. Most existing works inject latent variables into the Transformer decoder by the three paradigms, Embedding (Li et al., 2020a), Memory (Li et al., 2020a; Fang et al., 2021) and Softmax(Wang and Wan, 2019), as discussed in Sec. 1, while these methods shallowly fuse the latent variables with hidden states. To achieve deeper fusion and ameliorate KL vanishing, we propose DELLA.

The most relevant architecture to our model is hierarchical VAE (Sønderby et al., 2016; Klushyn

[1] DELLA: **DE**eply Fused **L**ayer-wise **LA**tent Variables

et al., 2019; Vahdat and Kautz, 2020; Child, 2020), which is mainly designed for image generation and not suitable for text. For text generation, hierarchical latent variables are either independent of each other (Serban et al., 2016), or corresponding to different text granularities (sentence or word level), while our DELLA learns conditionally inferred and layer-wise latent variables based on Transformer.

3 Preliminaries

3.1 Transformer

Transformer (Vaswani et al., 2017) represents an input sequence $\boldsymbol{x} = \{x_1, \ldots, x_i, \ldots, x_n\}$ as contextualized distributed hidden states $\boldsymbol{h} = \{h_1, \ldots, h_i, \ldots, h_n\}$ by a series of stacked layers, and states in the l-th layer, $\boldsymbol{h}^{(l)}$, are calculated with scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) V^T, \tag{1}$$

where Q, K, V stand for Query, Key, Value, respectively, which are projected from outputs of the previous layer: $Q = W^q \boldsymbol{h}^{(l-1)}$, $K = W^k \boldsymbol{h}^{(l-1)}$, $V = W^v \boldsymbol{h}^{(l-1)}$. d is the dimension of hidden states. In practice, multiple groups of states are calculated with different attention parameters and then concatenated, known as multi-head attention.

3.2 VAE

As a kind of generative model, VAE estimates the intractable data distribution $p(\boldsymbol{x})$ by deriving and maximizing its lower bound as:

$$\begin{aligned} &\log p(\boldsymbol{x}) \geq \mathcal{L}_{ELBO}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \\ &\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \end{aligned} \tag{2}$$

where $\boldsymbol{z}$ is the latent variable and $p(\boldsymbol{z})$ is the prior distribution of latent variable which is commonly assumed as standard Gaussian; the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ is approximated by an inference network (encoder) $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$; $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is a generator (decoder) to generate text $\boldsymbol{x}$ from the latent variable $\boldsymbol{z}$; $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are corresponding parameters.

The whole lower bound in Eq.(2), called Evidence Lower BOund (ELBO), consists of two terms: the reconstruction loss,

$$\mathcal{L}_E = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right], \tag{3}$$

which helps reconstruct the input given the posterior latent variable $\boldsymbol{z}$, and the KL divergence,

$$\mathcal{L}_R = \text{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})\right). \tag{4}$$

In practice, VAE is considered as a regularized Auto-encoder, and a hyper-parameter β is introduced to control the strength of KL, $\beta\mathcal{L}_R$, usually used in KL annealing methods (Fu et al., 2019).

3.3 Incorporate Transformer into VAE

For Transformer encoder, the posterior $\boldsymbol{z}$ is mapped from the text representation, which can be the pooling of all hidden states in the last layer (Fang et al., 2021), or state of a special token (Li et al., 2020a), *e.g.*, [CLS]. Then $\boldsymbol{z}$ is injected into Transformer decoder by the paradigms discussed in Sec. 1.

Now we take a further step and investigate why intrinsically these three paradigms, namely Embedding, Memory and Softmax, would perform poorly.

Embedding: Define $\boldsymbol{e}_i, \boldsymbol{e}_j$ as two token embeddings and $\alpha_{i,j}$ as the attention weight of i-th and j-th tokens. From Eq.(1), we have $\alpha_{i,j} = (W^q \boldsymbol{e}_i)^T (W^k \boldsymbol{e}_j) = \boldsymbol{e}_i^T (W^q)^T W^k \boldsymbol{e}_j$, which is further abbreviated as $\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle$. Such Embedding paradigm directly adds $\boldsymbol{z}$ to token embeddings as:

$$\begin{aligned} \alpha'_{i,j} &= \left[W^q(\boldsymbol{e}_i + \boldsymbol{z})\right]^T \left[W^k(\boldsymbol{e}_j + \boldsymbol{z})\right] \\ &= \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle + \langle \boldsymbol{e}_i, \boldsymbol{z} \rangle + \langle \boldsymbol{z}, \boldsymbol{e}_j \rangle + \langle \boldsymbol{z}, \boldsymbol{z} \rangle, \end{aligned} \tag{5}$$

where we can find that a redundant term, $\langle \boldsymbol{z}, \boldsymbol{z} \rangle$, is introduced, bringing extra noise for attention mechanism. Moreover, information in $\boldsymbol{z}$ could diminish with propagation through layers (Fig. 2), aggravating KL vanishing.

Memory: This paradigm treats $\boldsymbol{z}$ as an additional memory token and places it at the beginning of $\boldsymbol{x}$ to be attended by other tokens via attention. Nevertheless, as mentioned in Sec. 1, the powerful Transformer decoder may only rely on preceding decoded tokens. Consequently, with no explicit constraints (*e.g.*, auxiliary loss), such a memory token is more likely to be ignored by self-attention (Fig. 6 & 7), even exacerbating KL vanishing.

Softmax: This paradigm first adds $\boldsymbol{z}$ to the last-layer hidden states $\boldsymbol{h}$, and then projects $\boldsymbol{z} + \boldsymbol{h}$ into a logit vector $\boldsymbol{p} \in \mathbb{R}^v$ over the vocabulary, where v is vocab size. In this method, latent variables do not interact with hidden states until the last layer, which erodes the effect of latent variables (see Fig. 2).

4 Methodology

As demonstrated in Sec. 3, existing three paradigms make latent variables gradually diminish through layers, be ignored by self-attention or inadequately interact with hidden states, which would not mitigate but even worsen the KL vanishing problem.

To deeply fuse latent variables with the whole computation path of Transformer, we propose DELLA to learn a series of layer-wise posterior latent variables which are conditionally inferred in encoder, and injected into hidden states in decoder by low-rank tensor product. We present layer-wise latent variables in Sec. 4.1, describe the tensor product fusion in Sec. 4.2, give the theoretical verification of DELLA's effectiveness for ameliorating KL vanishing in Sec. 4.3, and then extend DELLA to Conditional VAE (CVAE) in Sec. 4.4.

4.1 Layer-wise Latent Variables

Different from previous work where only one latent variable $\boldsymbol{z}$ is calculated and shared by (Li et al., 2020a) or projected to (Fang et al., 2021) decoder layers, we involve a series of latent variables $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L\}$, where L is the number of Transformer layers. Then we reformulate the prior and posterior distributions as $p(\boldsymbol{z}) = \prod_{l=1}^{L} p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})$, $q(\boldsymbol{z}|\boldsymbol{x}) = \prod_{l=1}^{L} q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x})$, respectively, with each $\boldsymbol{z}_l$ still following Gaussian distribution. Then we rewrite $\mathcal{L}_R$ in Eq.(4) similar to Vahdat and Kautz (2020):

$$\begin{aligned}\mathcal{L}_R &= \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \\ &= \sum_{l=1}^{L} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})} \left[\mathrm{KL}(q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))\right].\end{aligned} \tag{6}$$

When $l = 1$, $p(\boldsymbol{z}_1|\boldsymbol{z}_{<1}) = p(\boldsymbol{z}_1)$ is the standard Gaussian distribution, $q(\boldsymbol{z}_1|\boldsymbol{x}, \boldsymbol{z}_{<1}) = q(\boldsymbol{z}_1|\boldsymbol{x})$. We give detailed derivations in Appendix B.1.

These latent variables $\boldsymbol{z}_l$ are calculated (inferred) layer by layer using representations of the corresponding layer. Concretely, in l-th layer, we use the hidden state of the first token in text $\boldsymbol{x}$, as its l-th-layer representation, denoted as $\boldsymbol{x}^{(l)} \in \mathbb{R}^d$, where d is hidden size. Then we represent latent variables in lower layers as $\boldsymbol{z}_{<l}$ and obtain it by:

$$\boldsymbol{z}_{<l} = \tanh(\boldsymbol{W}_{hh}^{(l)} \boldsymbol{z}_{<l-1} + \boldsymbol{W}_{ih}^{(l)} \boldsymbol{z}_{l-1}), \tag{7}$$

where $W_{hh}, W_{ih} \in \mathbb{R}^{p \times p}$, so $\boldsymbol{z}_{<l} \in \mathbb{R}^p$ and p is the dimension of latent variable. $\boldsymbol{z}_0$ and $\boldsymbol{z}_{<0}$ are set as zero vector. We calculate the mean and variance vectors of $p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})$ and $q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x})$ by:

$$\begin{aligned}\begin{pmatrix}\boldsymbol{\mu}_p \\ \log(\boldsymbol{\sigma}_p^2)\end{pmatrix} &= \boldsymbol{W}_p^{(l)} \boldsymbol{z}_{<l}, \\ \begin{pmatrix}\boldsymbol{\mu}_q \\ \log(\boldsymbol{\sigma}_q^2)\end{pmatrix} &= \boldsymbol{W}_q^{(l)} \begin{pmatrix}\boldsymbol{z}_{<l} \\ \boldsymbol{x}^{(l)}\end{pmatrix},\end{aligned} \tag{8}$$

where $\boldsymbol{W}_p \in \mathbb{R}^{p \times 2p}$, $\boldsymbol{W}_p \in \mathbb{R}^{p \times 2p}$.

The latent variable $\boldsymbol{z}_l$ is sampled from the posterior distribution $q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2 \boldsymbol{I})$ for training, and from the prior one $q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) = \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\sigma}_p^2 \boldsymbol{I})$ for testing. Since hidden states in each layer belong to different vector spaces, the parameters to calculate each $\boldsymbol{z}_{<l}$, *e.g.*, $\boldsymbol{W}_p^{(l)}$ and $\boldsymbol{W}_q^{(l)}$, *do not* share throughout different layers.

4.2 Low-rank Tensor Product

We inject the latent variable $\boldsymbol{z}_l$, which is obtained based on l-th encoder layer, into the corresponding l-th decoder layer. Instead of simply using $\boldsymbol{z}_l$ as a memory token as discussed in Sec. 3.3, we resort to low-rank tensor product, which has been successfully utilized for fusing multimodal representations (Liu et al., 2018), to deeply fuse latent variables with hidden states in the decoder.

In detail, we conduct low-rank tensor product on $\boldsymbol{z}_l$ and $\boldsymbol{x}_i$'s l-th-layer value vector $\boldsymbol{v}_i^{(l)}$ as:

$$\widetilde{\boldsymbol{v}}_i^{(l)} = (\sum_{j=1}^{r} \boldsymbol{W}_v^{(l,j)} \boldsymbol{v}_i^{(l)}) \circ (\sum_{j=1}^{r} \boldsymbol{W}_z^{(l,j)} \boldsymbol{z}_l), \tag{9}$$

where r is a hyper-parameter, $\circ$ means element-wise multiplication, $\boldsymbol{W}_v \in \mathbb{R}^{d \times d}$.$\boldsymbol{W}_z \in \mathbb{R}^{p \times d}$ are learnable parameters which are shared across all positions (i) but not shared with layers (l), considering distinct vector spaces in different layers, as mentioned in Sec. 4.1. Then the fused Value $\widetilde{V}^{(l)} = \{\widetilde{\boldsymbol{v}}_1^{(l)}, \ldots, \widetilde{\boldsymbol{v}}_n^{(l)}\}$ is used in Eq.(1)

In this way, layer-wise $\boldsymbol{z}_l$ is conditionally inferred from latent variables in previous encoder layers, together with l-th-layer text representation, and then explicitly fused with the corresponding decoder layer, yielding a deeper intervention throughout the whole computation path of Transformer.

4.3 Why Could DELLA Work Well?

To theoretically interpret the advantage of layer-wise latent variables which contributes most to DELLA (Table 4), we give the following theorem:

Theorem 1 *For an observation $\boldsymbol{x}$ and a sequence of latent variables $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots \boldsymbol{z}_L$, satisfying $p(\boldsymbol{z}) = \prod_{l=1}^{L} p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})$, and $q(\boldsymbol{z}|\boldsymbol{x}) = \prod_{l=1}^{L} q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x})$, then the expectation of the KL term, $\mathbb{E}_{p(\boldsymbol{x})}[\mathcal{L}_R]$ is an upper bound of:*

$$-\sum_{i=2}^{L-1} I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_i|\boldsymbol{z}_{i-1}) - I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_1|\boldsymbol{x}), \tag{10}$$

where I is the interaction information[2].

See Appendix B.2 for proof. Based on Theorem 1, minimizing $\mathcal{L}_R$ approximatively means maximizing each interaction information term in Eq.(10), which forces the entanglement of all latent varibles $\boldsymbol{z}_1; \ldots; \boldsymbol{z}_L$ given the observation $\boldsymbol{x}$, alleviating the diminishing of information encoded in latent variables when propagating through layers.

4.4 Extension to CVAE

DELLA could also be applied to CVAE for conditional generation tasks like storytelling. Given an observation $\boldsymbol{x}$ and its condition $\boldsymbol{c}$, we can optimize:

$$\begin{aligned}\log p(\boldsymbol{x}|\boldsymbol{c}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{c})] \\ - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})||p(\boldsymbol{z}|\boldsymbol{c})),\end{aligned} \tag{11}$$

and then replace the prior distribution $q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})$ and posterior distribution $p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})$ in Eq.(6) with $q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{z}_{<l})$ and $p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{c})$, respectively.

In this case, we encode the condition $\boldsymbol{c}$ with the same encoder. Similarly, we can obtain the representation of $\boldsymbol{c}$ at l-th layer, denoted as $\boldsymbol{c}^{(l)} \in \mathbb{R}^d$, and then calculate the mean and log variance of $p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{c})$ and $q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x}, \boldsymbol{c})$ by:

$$\begin{aligned}\begin{pmatrix}\boldsymbol{\mu}_p \\ \log(\boldsymbol{\sigma}_p^2)\end{pmatrix} &= \hat{\boldsymbol{W}}_p^{(l)} \begin{pmatrix}\boldsymbol{z}_{<l} \\ \boldsymbol{c}^{(l)}\end{pmatrix}, \\ \begin{pmatrix}\boldsymbol{\mu}_q \\ \log(\boldsymbol{\sigma}_q^2)\end{pmatrix} &= \hat{\boldsymbol{W}}_q^{(l)} \begin{pmatrix}\boldsymbol{z}_{<l} \\ \boldsymbol{x}^{(l)} \\ \boldsymbol{c}^{(l)}\end{pmatrix},\end{aligned} \tag{12}$$

where $\hat{\boldsymbol{W}}_p^{(l)} \in \mathbb{R}^{(p+d)\times 2p}$, $\hat{\boldsymbol{W}}_q^{(l)} \mathbb{R}^{(p+2d)\times 2p}$.

5 Experiment

5.1 Dataset

We consider four datasets for language modelling and unconditional generation, including the Yelp, and Yahoo (Yang et al., 2017; He et al., 2019), Penn Treebank (PTB) (Marcus et al., 1993), and SNLI (Bowman et al., 2015), and three datasets for conditional generation tasks, including summarization generation with CNN/DailyMail (CNN/DM) (See et al., 2017), story generation with WritingPrompts (WP) (Fan et al., 2018) and paraphrase generation with Quora [3]. Detailed data statistics are listed in Table 7. Due to the limited computation capability, we use 165,157 samples in CNN/DM and 22,2614 in WP with the max length of 900 for training.

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

[3]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

5.2 Implementation Details

We use pretrained language models as the backbone and fine-tune them on each task mentioned above with our DELLA as in (Li et al., 2020a). For unconditional generation and story generation, encoder and decoder **shared** the same parameters initialized with 12-layer GPT-2 (Radford et al., 2019). For summarization and paraphrase generation, parameters are **not shared** and initialized with BART-base (Lewis et al., 2020). We set the dimension of latent variable as 32 for all VAE-based models and use cyclical annealing for training, following (Li et al., 2020a). More details are given in Appendix A.1.

5.3 Baseline

We make a comprehensive comparison with strong Transformer-based baselines. We do not consider RNN-based models that are inferior to Transformer for text generation as shown in (Li et al., 2020a).

Finetuned Pretrained Models. To manifest the suitability of DELLA for different pretrained language models, we compare it with fine-tuned GPT2 on unconditional generation and story generation, and with fine-tuned BART-base on summarization generation and paraphrase generation.

Optimus (Li et al., 2020a): a large-scale VAE model which takes a pre-trained BERT as encoder and pretrained GPT-2 as decoder. This model is first pretrained as a VAE, which simultaneously utilizes the two paradigms, Embedding and Memory as introduced in Sec. 3.3, for injecting latent variables, with both KL annealing and KL threshold tricks, and then fine-tuned on downstream tasks.

Transformer-based VAE. Besides Optimus, we also compre the three paradigms, namely **Embedding** (Li et al., 2020a), **Memory** (Fang et al., 2021) and **Softmax** (Wang and Wan, 2019), and incorporate each paradigm into the same pre-trained model as DELLA on each dataset for fair comparison.

5.4 Metrics

For unconditional generation tasks, we consider three types of metrics. **(a) Representation Learning Capability**: we report PPL, ELBO, KL, mutual information (MI) (Alemi et al., 2016) and activate units (AU) (Burda et al., 2016). These metrics measure VAE's ability to mitigate KL vanishing and learn meaningful representations. Different from traditional language models like GPT-2, VAE-based models could not produce exact PPL due to randomness, so we use importance-weighted sam-

Model	Representation Learning					Generation Quality			Generation Diversity		
	PPL↓	ELBO↓	KL↑	MI↑	AU↑	BLEU↑	CND↓	MAUVE↑	SB↓	Dist↑	JS ↓
Dataset: Yelp											
GPT-2	22.13	-	-	-	-	56.92	0.68	0.12	65.90	**17.96**	0.51
Optimus	22.79	344.10	15.09	7.67	-	-	-	-	-	-	-
Embed	19.98	327.28	4.77	4.14	6	56.34	0.31	0.42	65.27	15.59	0.44
Memory	19.95	326.60	5.70	5.30	11	**57.37**	0.27	0.46	63.90	16.91	**0.39**
Softmax	20.14	328.13	7.50	6.29	13	56.83	0.30	0.45	64.26	16.51	0.40
DELLA	**12.35**	**239.83**	**29.47**	**10.78**	**23**	57.15	**0.13**	**0.55**	**60.02**	17.63	0.43
Dataset: Yahoo											
GPT-2	24.17	-	-	-	-	44.25	0.55	0.15	54.06	21.07	**0.28**
Optimus	23.11	293.34	17.45	8.85	-	-	-	-	-	-	-
Embed	22.18	286.85	3.63	3.03	3	42.27	0.45	0.31	54.15	20.80	0.32
Memory	22.03	285.47	4.87	4.62	18	**45.20**	0.46	0.37	54.59	21.87	0.33
Softmax	22.35	287.44	6.35	5.52	19	44.28	0.44	0.34	54.49	21.65	0.32
DELLA	**11.49**	**201.34**	**27.84**	**12.31**	**21**	44.67	**0.19**	**0.38**	**48.53**	**21.88**	0.31

Table 1: Evaluation results for language modelling and unconditional generation. Results of Optimus are directly copied from the original paper with $\lambda = 0.5$. SB means Self-BLEU.

ples to estimate PPL, following He et al. (2019). We set the threshold in AU to 0.2 to further distinguish different models. **(b) Generation Quality**: we report BLEU (Papineni et al., 2002), CND (Li et al., 2020b) and MAUVE (Pillutla et al., 2021). CND and MAUVE measure the divergence between human-authored text and the generated one. **(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 assess the diversity and novelty of generated text.

For conditional generation tasks, we report BLEU, Rouge-1, Rouge-2, Rouge-L (Lin and Hovy, 2002), and BERTScore (Zhang et al., 2020) to evaluate the quality of generated texts, as well as the same diversity metrics used in unconditional generation. We also report KL and AU value to present representation learning capability. More details of metrics are provided in Appendix A.3.

5.5 Results

5.5.1 Unconditional Generation

We present results on Yelp and Yahoo in Table 1 and leave the those on PTB and SNLI in the Appendix A.5 due to space limitations. We also show the learning curves of ELBO and KL in Fig. 5.

As shown in Table 1, DELLA achieves notably improvement on almost all the metrics, especially superior on representation learning metrics. Much higher KL, MI and AU, and a big gap in PPL obtained by DELLA indicate the latent variables encode more meaningful text information and won't diminish when propagating through Transformer layers, which strongly supports our motivation that fusing latent variables with hidden states more deeply could effectively alleviate the KL vanishing problem. Such results also empirically verify the theoretical advantage of our model (Theorem 1), demonstrating entangled layer-wise latent variables can preserve more encoded knowledge for decoder. We will show that $\boldsymbol{z}$ can involve more information when injected into more layers in Sec. 5.8.

Besides, DELLA also gets good performance (comparable BLEU and much better CND and MAUVE) on generation quality. With more informative latent variables, DELLA could achieve a better ELBO and hence further boost the learning of data distribution $p(\boldsymbol{x})$ in Eq.(2), leading to satisfactory quality of generated texts.

Generally, DELLA also outperforms baseline models on generation diversity. The reason is twofold: randomly sampled latent variables $\boldsymbol{z}$ should bring diversity, while the VAE-based baselines tend to ignore $\boldsymbol{z}$ as mentioned before, losing some randomness. In contrast, latent variables are deeply fused in DELLA, maintaining enough randomness. Besides, each latent variable is sampled in corresponding layer, and thus such a sampling process accumulates and enhances randomness, further benefiting diversity while keeping good quality.

5.5.2 Conditional Generation

We report the results of WP and CNN/DM in Table 2, and leave those of Quora in Appendix A.5. As we can see, DELLA performs better on most quality metrics, but gets a little worse on diversity

Model	Quality					Diversity			KL↑	AU↑
	BLEU↑	Rouge-1↑	Rouge-2↑	Rouge-L↑	BERTScore↑	SB↓	Dist↑	JS ↓		
Dataset: WritingPrompts										
GPT-2	27.89	27.72	7.96	14.30	78.12	**53.78**	**22.99**	**0.51**	-	-
Embed	39.67	**36.17**	7.96	15.78	81.64	64.55	14.31	0.73	2.35	3
Memory	40.79	36.13	8.04	16.16	81.68	67.56	12.90	0.80	0.07	0
Softmax	41.04	36.14	8.12	16.30	81.75	67.02	13.08	0.78	0.32	0
DELLA	**41.39**	35.46	**8.78**	**17.20**	**81.77**	56.28	20.91	0.60	**28.14**	**8**
Dataset: CNN/DM										
Bart-base	48.74	**41.33**	19.82	29.63	87.75	29.94	43.68	0.10	-	-
Embed	44.10	40.43	19.41	29.43	87.60	29.60	44.04	0.10	0.0	0
Memory	46.02	41.18	19.74	29.64	87.78	29.79	43.92	0.11	0.0	0
Softmax	44.40	40.94	19.63	29.61	87.00	29.64	44.11	0.10	0.0	0
DELLA	**49.18**	41.27	**19.85**	**29.84**	**88.09**	**29.07**	**44.24**	**0.09**	**0.91**	**1**

Table 2: Evaluation results for conditional generation.

Dataset: WritingPrompts			
Model	Fluency	Coherence	Novelty
GPT2	1.83	2.12	2.50
Embed	2.16	2.33	2.67
Memory	2.45	2.28	2.78
Softmax	2.48	**2.42**	2.85
DELLA	**2.51**	2.38	**2.89**
Dataset: CNN/DM			
Model	Informativeness	Coherence	Novelty
Bart-base	**3.12**	4.32	3.52
Embed	2.88	4.08	3.50
Memory	2.95	4.23	3.48
Softmax	2.91	**4.33**	3.50
DELLA	3.05	**4.33**	**3.56**

Table 3: Human evaluation results on conditional generation. The scores range from 1 (worst) to 5 (best). The p-value is 0.002 and Kappa score is 0.64 which indicates acceptable inter-annotator agreement.

compared to GPT-2. This is because GPT-2 may produce some ill-formed contents which 'improve' diversity by cheating the metrics but also lead to much worse quality (lower BLEU and Rouge). Even so, on both WP and CNN/DM, DELLA still beats all previous VAE paradigms in diversity, manifesting the effectiveness of our DELLA.

In addition, all baselines methods suffer from severer KL vanishing problems on conditional generation tasks than on the unconditional ones. This is because the given condition text could aggravate the reliance of these models on preceding generated tokens and the condition, and therefore bypass latent variables. By contrast, DELLA could learn more informative $\boldsymbol{z}$ and hence keep a relatively higher KL value even given the condition text.

Model	PPL↓	ELBO↓	KL↑	MI ↑	AU↑
DELLA	**12.35**	**239.83**	**29.47**	**10.78**	**23**
-LTP	12.68	249.32	28.52	9.77	21
-LW	19.88	324.45	20.12	7.23	18
Separate	14.17	286.30	28.82	9.88	16
$l = 1$ KL	12.55	266.97	0.15	0.15	0
$l = 12$ KL	12.48	263.38	0.73	0.61	0
Embed(384)	20.11	327.29	0.55	0.38	0
Memory(384)	20.09	326.24	0.46	0.25	0
Softmax(384)	20.15	330.24	5.04	7.15	0

Table 4: Ablation study on Yelp dataset. LTP: low-rank tensor product. LW: layer-wise latent variables. Separate: latent variables in each layer are independent. $l = 1$ or 4 KL means we only compute KL loss on $\boldsymbol{z}_1$ or $\boldsymbol{z}_L$, respectively. 384 means the dimension of latent variable used in baseline are $12 \times 32 = 384$.

5.6 Human Evaluation

To better verify the effectiveness of DELLA, we also conduct human evaluation on the two conditional generation tasks. For each model, we generated 30 samples on each task, and invite 5 competent annotators to score these samples in terms of three criteria, **Fluency**, **Coherence** and **Novelty** for story generation, and **Informativeness**, **Coherence** and **Novelty** for summarization generation.

As shown in Table 3, DELLA obtains satisfactory performance in quality, and is consistently superior to all baselines on diversity and novelty. See Appendix A.4 for more detailed evaluation protocols.

5.7 Ablation Study

Table 4 shows the results of ablation study on Yelp. We can find both tensor product and the layer-wise latent variables benefit the learning of informative

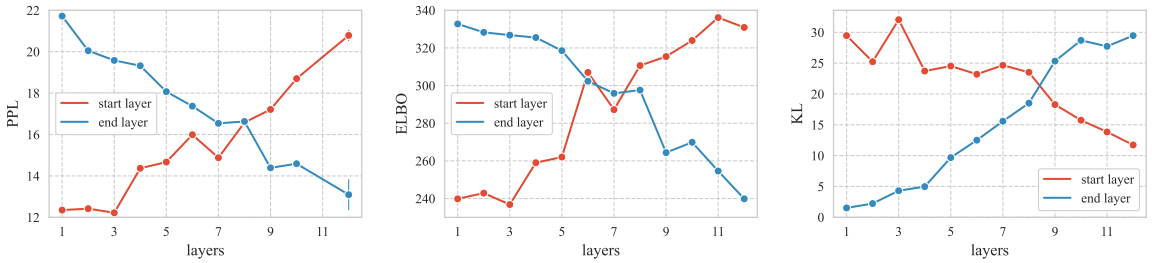

Figure 2: PPL, ELBO end KL on Yelp with different numbers of latent variables. The values *start layer* i and *end layer* j means latent variables are produces and utilized only from i-th layer to the last layer, or from the first layer to j-th layer of the encoder respectively.

Model	PPL↓	ELBO↓	KL↑	MI ↑	AU↑
Embed	22.21	339.12	0.03	0.03	0
+BOW	19.98	326.51	2.75	2.48	4
+Annealing	19.98	327.28	4.77	4.14	6
+Annealing + BOW	20.59	332.44	19.51	9.12	**28**
+Annealing + BN	21.14	338.59	21.09	8.98	25
Memory	22.16	338.68	0.00	0.01	0
+BOW	19.87	326.00	3.89	3.59	8
+Annealing	19.95	326.60	5.70	5.30	11
+Annealing + BOW	20.41	331.09	18.76	9.14	**28**
+Annealing + BN	20.25	331.59	18.11	9.07	24
Softmax	22.43	333.93	0.47	0.3	0
+BOW	20.53	331.89	10.16	5.57	**28**
+Annealing	20.14	328.13	7.50	6.29	13
+Annealing + BOW	21.14	335.48	17.51	8.46	**28**
+Annealing + BN	20.95	337.10	21.25	9.15	25
DELLA	17.18	312.45	9.39	5.32	6
+BOW	13.98	289.94	11.59	9.25	8
+Annealing	**12.35**	**239.83**	29.47	10.78	23
+Annealing+BOW	12.82	249.98	**32.79**	**11.26**	26
Backbone: GPT-2 medium (24 layers)					
Embed	18.33	317.44	2.13	1.44	3
Mem	18.30	317.24	4.47	4.26	10
Softmax	18.47	318.80	5.80	5.03	12
DELLA	**11.01**	**230.96**	**17.09**	**23.69**	**27**

Table 5: Results on Yelp for transformer-bsaed VAE with BOW loss, KL annealing and batch normalization tricks, and use 24-layer GPT2-medium as backbone. Here we fix γ in batch normlization as 1.

latent variables, while the latter contributes the most to DELLA. To further verify the performance gain originating from Theorem 1 instead of simply increasing the number or the dimension of latent variables, we conduct two groups of experiments.

First, we remove the conditional dependence between layer-wise latent variables by independently sampling each $\boldsymbol{z}_l$ in both training and testing. We can see that removing dependence causes a significant performance drop. Besides, we keep the dependence between $\boldsymbol{z}_l$ but optimize only one of the KL terms in Eq.(6), and find all representation capability metrics deteriorate, especially KL, MI and AU. Such results effectively demonstrate the necessity of using and optimizing the conditional inference of layer-wise latent variables, supporting our theoretical interpretation of DELLA.

Second, we enlarge the dimension of $\boldsymbol{z}_l$ used in the three paradigms to 384 (12×32), equal to the total latent dimension used in DELLA. The results show that simply increasing the dimension of latent variables brings a more sparse latent space, even exacerbating the KL vanishing problem.

5.8 Analysis

Training Tricks To reveal the robustness of our model, we evaluate the influence of three commonly used training tricks to relieve KL vanishing, *i.e.*, BOW (bag-of-words) loss (Wang et al., 2017), batch normalization (Zhu et al., 2020) and KL annealing (Fu et al., 2019), to the performance of DELLA and the three paradigms. As shown in Table 5, previous methods suffer KL vanishing seriously without annealing or BOW loss, getting KL, MI and AU almost 0. Though not good as using annealing, DELLA still maintains acceptable performance and mitigates KL vanishing even without any training tricks. Bow and batch normalization dramatically prevent low KL divergence, but obstruct the optimization and thus cause higher PPL.

Number of Latent Variables We observe the change of PPL, KL and ELBO with different numbers of latent variables. We conduct two groups of experiments where we produce and utilize layer-wise latent variables starting from and ending at different layers. As shown in Fig. 2, incorporating more latent variables could continuously improve performance, consistent to our claim in Sec. 4. With the same number of latent variables, starting from a higher layer is better than ending at a lower layer, which indicates that latent variables

Sentence 1: A man is taking beautiful photos by the river
0.1: A man is taking photos by the beautiful river
0.3: A man is taking **pictures** by the river
0.5: A man is enjoying the **picnic** by the river
0.7: A goat is enjoying the **day** by the river
0.9: A goat is enjoying the **day** on the farm
Sentence 2: A goat is enjoying the sun on the farm

Sentence 1: Two girls are playing baseball on the beach
0.1: Two girls are playing **basketball** on the beach.
0.3: Two girls are playing **music** in the **park**.
0.5: Two girls are **singing** on the beach.
0.7: **Crowds of** people are dancing in the **park**
0.9: Lots of people are dancing in the **park**
Sentence 2: Many people are dancing in front of building

Figure 3: Interpolating latent space. The sentence in each row is generated with a latent variable interpolated from those of sentence 1 and sentence 2.

Prompt: After receiving perfect score on your final exam, your school is asking you to give a speech on the subject of the exam. Little does anyone know you guessed every single answer.

Memory: The room was silent except for me. I could hear some whispering behind us as we continued to pace about campus, but there was no need. We knew better than that any moment that would come, and at least if they were around, we wouldn't have to be so paranoid.

DELLA: "Hey mister!" I just got back from work and the whole class was already ready for question. So go ahead I **took the stage**. The teacher **smiled encouragingly**. Standing up in the front of desk, I started talking about all sorts of things about **answering test question**.

Figure 4: Generation examples of Memory and DELLA based on the prompt from test set of WritingPrompts.

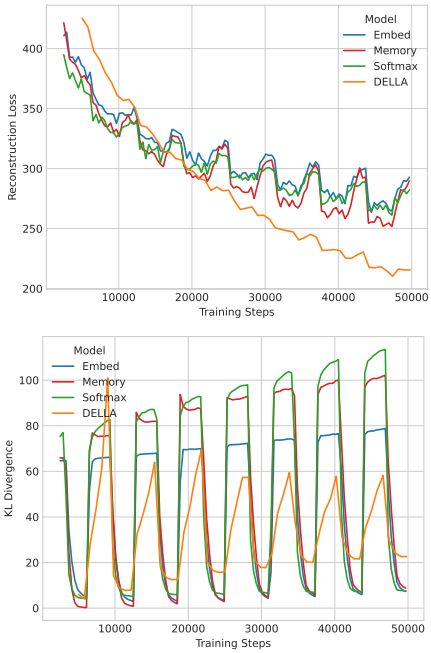

Figure 5: Reconstruction loss and KL Divergence throughout training process.

generated from higher layers encode more helpful information compared to those from lower layers, manifesting disadvantages of the two previous paradigms, Softmax (starting from the last layer) and Embedding (ending at the first layer).

Model size We compare the performance of DELLA and three paradigms with 24-layer GPT2-medium as backbone. As shown in Table 5, with the increasing of model size, DELLA consistently achieves better performance than baselines.

5.9 Case Study

VAE captures text representations in a smooth latent space. We take two sentences $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and sample two posterior latent variables $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$ from $p(\boldsymbol{z}^{(1)}|\boldsymbol{x}_1)$ and $p(\boldsymbol{z}^{(2)}|\boldsymbol{x}_2)$, and get interpolated latent variables with $\boldsymbol{z} = \tau\boldsymbol{z}^{(1)} + (1-\tau)\boldsymbol{z}^{(2)}$. We generate multiple sentences with a continuously changed τ from 0 to 1. As shown in Fig. 3, sentences generated from interpolated $\boldsymbol{z}$ mix the semantics of the two initial sentences and smoothly change from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$, showing DELLA's ability of learning a flexible latent space.

Fig. 4 shows the generation examples of DELLA and one of baseline, Memory, given the same prompt WritingPrompts. We observe that the generated text of Memory is irrelevant to the prompt, while DELLA generates coherent and vivid text.

6 Conclusion

In this paper, we propose a novel variational Transformer framework DELLA. Our framework learns a series of layer-wise latent variables with iterative dependence. These latent variables are conditionally inferred and injected into corresponding decoder layers by low-rank tensor product for deeper fusion. The experiments on both unconditional and conditional generation tasks demonstrate DELLA's ability to significantly mitigate KL vanishing and improve generated text's quality and diversity. In the future, we plan to explore further the potential of DELLA in larger pretrained language models.

Acknowledgement

Thanks for the anonymous reviewers for their comments. This work is supported by the National Key R&D Program of China (No. 2020AAA0106502) and International Innovation Center of Tsinghua University, Shanghai, China.

References

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

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*.

Rewon Child. 2020. Very deep vaes generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*.

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.

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*.

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*.

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.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2017. A deep generative framework for paraphrase generation. In *National Conference on Artificial Intelligence*.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5880–5894, Florence, Italy. Association for Computational Linguistics.

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*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

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

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751.

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. 2019. Learning hierarchical priors in vaes. In *Neural Information Processing Systems*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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 *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 3603–3614, Hong Kong, China. Association for Computational Linguistics.

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.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical Chinese poems via conditional variational autoencoder and adversarial training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3890–3900, Brussels, Belgium. Association for 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: Computation and Language*.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, AmirAli Bagher Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256, Melbourne, Australia. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Neural Information Processing Systems*.

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

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

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.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, Copenhagen, Denmark. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv: Computation and Language*.

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*.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. 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*.

Liwei Wang, Alexander G. Schwing, and Svetlana Lazebnik. 2017. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *Neural Information Processing Systems*.

Tianming Wang and Xiaojun Wan. 2019. T-cvae: transformer-based conditioned variational autoencoder for story completion. In *International Joint Conference on Artificial Intelligence*.

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.

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.

Meng-Hsuan Yu, Juntao Li, Danyang Liu, Dongyan Zhao, Rui Yan, Bo Tang, and Haisong Zhang. 2020. Draft and edit: Automatic storytelling through multi-pass 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 Implementation Details

We load pretrained model GPT-2 (Radford et al., 2019) as initial parameters for unconditional generation and story generation, and pretrained BART-base (Lewis et al., 2020) for summarization and paraphrasing generation tasks. For the summarization and paraphrasing generation, we keep the encoder-decoder attention block. No encoder-decoder attention is used in unconditional generation and story generation tasks. The number of layers and dimensions of hidden states in DELLA is consistent with the configurations of corresponding pretrained models (GPT-2 has 12 layers and Bart-base has 6-layer encoder and 6-layer decoder. The hidden size of both is 768). We use the state of a special token to obtain the representation in the encoder. We utilize cyclical annealing tricks to train DELLA and other VAE baselines. Specifically, two epochs are one annealing period. In one period, β (the weight of KL term in ELBO) keeps 1e-5 in the first half, then linearly increases to 1 in the next quarter, then keeps at 1 for the last quarter. We select batch size over $\{16, 32\}$ and learning rate over {5e-5, 7e-5}. We use beam search for DELLA and top-k sampling for compared baseline models for the unconditional generation and story generation. For the summarization and paraphrasing generation, we use beam search in all the models.

We implement DELLA and other VAE baselines based on Huggingface Transformers (Wolf et al., 2020) library of v4.10.0 and use NVIDIA GeForce RTX 3090 to train our model. The total number of training GPU hours on different datasets is in Table 6. The number of parameters for our model is 193,353,984 in the unconditional generation setting and 195,180,114 in the conditional generation one. All experimental results are trained and tested in a single run.

Dataset	Training Time
Yelp	20h
Yahoo	20h
PTB	6h
SNLI	12h
CNN/DM	40h
WP	170h
Quora	5h

Table 6: GPU hours of training DELLA with RTX3090

Dataset	# Train	# Dev	# Test	Avarage Length	
Yelp	100k	10k	10k	96	
Yahoo	100k	10k	10k	79	
PTB	42k	3k	3k	21	
SNLI	100k	10k	10k	10	
CNN/DM	287k	13k	11k	S: 790	T: 61
WP	272k	15k	15k	S: 28	T: 674
Quora	134k	5k	10k	S: 10	T: 10

Table 7: **Statistics of datasets**. We present the size of train/dev/test sets and the average length for 7 datasets. S means source text and T means target text.

A.2 Datasets Details

The detailed dataset statistics are in Table 7. For the licenses of the datasets we use, CNN/DM and WritingPrompts use MIT License, while SNLI uses CC BY-SA 4.0. Meanwhile, PTB, Quora, and Yelp use their own license: LDC User Agreement, Yelp Data Agreement, and Quora's Terms of Service, respectively. All of these licenses and agreements allow their data for academic use. Unfortunately, we did not find the license for the Yahoo Dataset.

A.3 Metrics Details

Here we provide more details of the metrics used in our experiments.

Perplexity (PPL). PPL $= p(\boldsymbol{x})^{-1/n}$ is commonly used to evaluate the performance of language models, where n is number of tokens $\boldsymbol{x}$ contains. For VAE-based model, we can only obtain the lower bound of $\log p(\boldsymbol{x})$. We consider k latent variables $z_1, z_2, \ldots, z_k$ sampled from the posterior distribution $q(\boldsymbol{z}_i|\boldsymbol{x})$. Based on the fact that average importance weights are an unbiased estimator of $\log p(\boldsymbol{x})$ (Burda et al., 2016) and Jensen's Inequality, we have:

$$\begin{aligned}\mathcal{L}_k &= \mathbb{E}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(\boldsymbol{x},\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x})}\right] \\ &\leq \log \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k}\frac{p(\boldsymbol{x},\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x})}\right] = \log p(\boldsymbol{x}).\end{aligned} \tag{13}$$

We use $\mathcal{L}_k$ to estimate $\log p(\boldsymbol{x})$ and calculate PPL.

ELBO. The ELBO is the sum of reconstruction loss and KL divergence.

KL. The KL divergence of the posterior and prior distribution.

Mutual Information(MI) (Alemi et al., 2016).

Mutual Information $\mathcal{I}(\boldsymbol{x}, \boldsymbol{z})$ is defined as:

$$\begin{aligned} &\mathcal{I}_q(\boldsymbol{x}, \boldsymbol{z}) \\ =&\mathbb{E}_{p(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} \log q(\boldsymbol{z}|\boldsymbol{x}) - \mathbb{E}_{q(\boldsymbol{z})} \log q(\boldsymbol{z}) \end{aligned} \tag{14}$$

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

Activate Units(AU) (Burda et al., 2016). AU is the active units in latent varibles, defined as $A_{\boldsymbol{z}} = \text{Cov}_{\boldsymbol{x}}(\mathbb{E}_{z\sim q(z|\boldsymbol{x})}[z]) > \delta$, where δ is a threshold, commonly set as 0.01. However, we find that with $\delta = 0.01$, all VAE models in our experiments have full active unit. So we increase the threshold to 0.2 to distinguish the performance of different models on this metric. **Please note** that DELLA incorporates latent variables in all layers, and hence we calculate AU for the latent variable in each layer and then report the average.

BLEU (Papineni et al., 2002). BLEU measures the n-gram overlap of generated sequences and the reference ones. For unconditional setting, we regard all samples in the test set as references to each generated example.

CND (Li et al., 2020b). CND approximates the divergence of the empirical reference distribution and generated text distribution in n-gram spaces.

MAUVE (Pillutla et al., 2021). MAUVE measures the gap between reference text and generated text using divergence frontiers.

Self-BLEU (Zhu et al., 2018). Self-Bleu calculates the BLEU score on the generated samples, which averages the BLEU score of each generated sequence calculated with other generated ones as references. This metric measures the diversity of a set of generated sequences. Higher Self-BLEU means these generated sequences are more distinguishable from each other.

Dist (Li et al., 2016). Dist measures the proportion of distinct n-grams on generated samples.

Jaccard Similarity(JS) (Wang and Wan, 2018). JS calculates the average n-gram Jaccard similarity between every two generated sequences.

Rouge (Lin and Hovy, 2002). Rouge computes n-gram overlap of generated examples with given target samples. We use rouge-score v0.0.4 to evaluate the rouge score of our model and the baselines.

BERTScore (Zhang et al., 2020). BERTScore uses pre-trained BERT (Devlin et al., 2019) to obtain the vector representations of generated and reference text and calculates their cosine similarity. We use bert-score v0.3.10 to calculate the BERTScore of our model and the baselines.

A.4 Human Evaluation Details

Due to the relatively long length of generated text, we randomly sample 30 examples in the test set of WP and CNN/DM as input to DELLA and other compared baseline models to generate the target. We invite five graduate students proficient in English to score the generated text. The criteria for story generation include fluency, coherence, and novelty, and the criteria for summarization generation include informativeness, consistency, and novelty. Specifically, fluency measures whether the generated sentences are syntactically fluent; coherence measures whether the generated text is logically structured and consistent with the input text; novelty measures whether the content is novel and attractive; informativeness measures to what extent the generated summarization summarizes the general idea of the article.

When conducting the human evaluation, we informed the participants as follows:

- The following contents are generated by the automatic models. Some of them may be offensive or contain improper arguments. Please be conscious of these risks and evaluate these contents equitably and adequately.
- The evaluation you provide will be used only for academic use and will never be used commercially.

Every evaluator will sign their signature below these warnings to confirm that they have read those words. After finishing the annotation, they will receive $25. This amount is determined by the time of the whole annotation process and the estimation of average hourly income. The ethics review board for data collection protocol is not essential in our country, so we did not conduct this review for our data collection protocol.

A.5 Additional Experimental Result

Table 8 and Table 9 report the results on PTB, SNLI and Quora dataset.

A.6 Case Study Details

We take two sentences $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and sample two groups of latent variables $\boldsymbol{z}^{(1)} = \{\boldsymbol{z}_1^{(1)}, \boldsymbol{z}_2^{(1)}, \ldots, \boldsymbol{z}_L^{(1)}\}$ and $\boldsymbol{z}^{(2)} = \{\boldsymbol{z}_1^{(2)}, \boldsymbol{z}_2^{(2)}, \ldots, \boldsymbol{z}_L^{(2)}\}$ from posterior distributions $p(\boldsymbol{z}^{(1)}|\boldsymbol{x}_1)$ and $p(\boldsymbol{z}^{(2)}|\boldsymbol{x}_2)$. We obtain the weighted latent variables $\hat{\boldsymbol{z}} = \{\hat{\boldsymbol{z}}_1, \hat{\boldsymbol{z}}_2, \ldots, \hat{\boldsymbol{z}}_L\}$ by

taking weighted sum at each corresponding element in two groups, i.e. $\hat{\boldsymbol{z}}_i = \tau * \boldsymbol{z}_i^{(1)} + (1-\tau) * \boldsymbol{z}_i^{(2)}$. The mixed sentence $\hat{\boldsymbol{x}}$ is generated conditioned on $p(\hat{\boldsymbol{x}}|\hat{\boldsymbol{z}})$ by the decoder.

A.7 Potential Risks and Limitations of our work

Due to the unclean corpus (especially in the WP dataset) we use where slang repeatedly appears, the model training on this corpus may also output some rude expressions during generation. Also, the text generated in the unconditional generation task is not controllable, which may contain some bias or politically sensitive expression. Besides, since our model significantly improves the quality and diversity of generated, it can produce more plausible texts like news, which could be possibly utilized to create fake news or disinformation. However, on the other hand, our model could benefit fairness in language generation. Previous text generation models tend to produce biases like gender or nationality biases, which means only the majority would be appropriately described while the minority may be ignored. These biases are mainly caused by the biased training corpus. With the same data, our model can improve the diversity of generated text, which is also potential for mitigating these biased. We will try to develop debiased language generation systems in future work to avoid these risks harming society.

While DELLA shows good performance on text generation, it has one limitation: training efficiency. DELLA brings more parameters compared with three baseline methods. Training efficiency needs to be considered if we further explore the performance of DELLA on the large pretrained model.

Model	Representation Learning					Generation Quality			Generation Diversity		
	PPL↓	ELBO↓	KL↑	MI↑	AU↑	BLEU↑	CND↓	MAUVE↑	SB↓	Dist↑	JS ↓
Dataset: PTB											
GPT-2	25.80	-	-	-	-	27.91	1.12	**0.73**	41.55	37.79	0.30
Optimus	22.79	344.10	15.09	7.67	-	-	-	-	-	-	-
Embed	19.98	327.28	4.77	4.14	6	28.04	1.38	0.69	41.32	34.46	0.33
Memory	24.41	90.25	1.22	1.17	4	21.31	1.21	0.58	26.58	38.28	**0.08**
Softmax	24.04	90.63	2.13	1.89	21	**28.59**	1.39	0.72	42.15	33.91	0.30
DELLA	**10.28**	**58.43**	**12.46**	**12.35**	**22**	28.15	**0.63**	0.68	**24.87**	**41.84**	0.17
Dataset: SNLI											
GPT-2	20.19	-	-	-	-	**63.57**	1.95	0.71	75.34	19.11	0.58
Optimus	16.67	38.50	16.35	8.89	-	-	-	-	-	-	-
Embed	13.79	32.97	3.24	3.16	20	59.26	0.98	**0.72**	65.59	20.89	0.44
Memory	13.78	32.62	2.13	2.08	10	62.80	1.24	0.67	54.59	21.87	0.33
Softmax	14.21	33.18	2.70	2.65	16	60.51	1.94	0.71	71.84	18.59	0.57
DELLA	**5.13**	**10.23**	**5.86**	**16.58**	**23**	62.94	**0.85**	0.69	**36.85**	**32.61**	**0.21**

Table 8: Additional results for language model and unconditional generation task. The results of Optimus are copied from original paper with $\lambda = 0.5$.

Model	BLEU↑	Rouge-1↑	Rouge-2↑	Rouge-L↑	Bertscore↑	KL↑
Bart-base	64.34	63.27	39.83	60.28	94.72	-
Embed	63.94	63.12	39.42	60.22	94.66	0.0
Mem	63.78	62.86	39.18	59.96	94.65	0.0
Softmax	64.30	63.25	39.92	60.39	94.71	0.0
DELLA	**64.40**	**63.80**	**40.58**	**61.03**	**94.84**	**3.88**

Table 9: Results on Quora dataset. Because the sentences in Quora are quite short and constrained, the results of the three diversity metrics on all baselines are almost the same. So we omit them here.

B Additional Proof

B.1 Derivation of KL Divergence of Layer-Wise Latent Variables

KL divergence of layer-wise latent variables

$$
\begin{aligned}
& KL(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \\
=& \int q(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} \\
=& \int \prod_{l=1}^{L} q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l}) \log \frac{\prod_{l=1}^{L} q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})}{\prod_{l=1}^{L} p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{z}_1 \, \mathrm{d}\boldsymbol{z}_2 \ldots \mathrm{d}\boldsymbol{z}_L \\
=& \sum_{i=1}^{L} \int \prod_{l=1}^{L} q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{z}_1 \, \mathrm{d}\boldsymbol{z}_2 \ldots \mathrm{d}\boldsymbol{z}_L \\
=& \sum_{l=1}^{L} \int q(\boldsymbol{z}_{<l}|\boldsymbol{x}) q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{z}_1 \, \mathrm{d}\boldsymbol{z}_2 \ldots \mathrm{d}\boldsymbol{z}_l \\
=& \sum_{l=1}^{L} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})} \mathrm{KL}(q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))
\end{aligned} \tag{15}
$$

B.2 Proof of Theorem 1

First, we consider on term in the summation and can obtain:

$$
\begin{aligned}
& \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})} [\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))] \\
=& \int q(\boldsymbol{x}) q(\boldsymbol{z}_{<l}|\boldsymbol{x}) q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z_l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} \\
=& \int q(\boldsymbol{x}, \boldsymbol{z}_l, \boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z_l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} \\
=& \int q(\boldsymbol{x}, \boldsymbol{z}_l, \boldsymbol{z}_{<l}) \log \Big(\frac{q(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{z}_{<l})}{q(\boldsymbol{x}|\boldsymbol{z}_{<l}) q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \frac{q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \Big) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} \\
=& \int q(\boldsymbol{z}_{<l}) q(\boldsymbol{x}, \boldsymbol{z}_l|\boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{z}_{<l})}{q(\boldsymbol{x}|\boldsymbol{z}_{<l}) q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} + \\
& \int q(\boldsymbol{x}|\boldsymbol{z}_l, \boldsymbol{z}_{<l}) q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) q(\boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} \\
=& \int q(\boldsymbol{x}, \boldsymbol{z}_l|\boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{z}_{<l})}{q(\boldsymbol{x}|\boldsymbol{z}_{<l}) q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{z}_l + \\
& \int q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) q(\boldsymbol{z}_{<l}) \log \frac{q(\boldsymbol{z}_l|\boldsymbol{z}_{<l})}{p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})} \, \mathrm{d}\boldsymbol{z}_l \, \mathrm{d}\boldsymbol{z}_{<l} \\
=& H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) - H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x}) + \mathbb{E}_{q(\boldsymbol{z}_{<l})} \mathrm{KL}(q(\boldsymbol{z}_l|\boldsymbol{z}_{<l}||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l})) \\
\geq& H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) - H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x})
\end{aligned} \tag{16}
$$

where H is the Shannon entropy. Then, the summation has a lower bound:

$$
\begin{aligned}
& \sum_{i=1}^{L} \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})} [\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))] \\
\geq& \sum_{i=1}^{L} H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}) - H(\boldsymbol{z}_l|\boldsymbol{z}_{<l}, \boldsymbol{x}) \\
=& H(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L) - H(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L|\boldsymbol{x}) \\
=& I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_L)
\end{aligned} \tag{17}
$$

where I is mutual information. Next, we prove the following inequality with induction:

$$I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_L) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_L) \tag{18}$$

When $L = 2$, we proof $I(\boldsymbol{x}; \boldsymbol{z}_1, \boldsymbol{z}_2) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \boldsymbol{z}_2)$. Actually, we have the following facts:

$$\begin{aligned} & I(\boldsymbol{x}; \boldsymbol{z}_1, \boldsymbol{z}_2) \\ =& H(\boldsymbol{x}) + H(\boldsymbol{z}_1, \boldsymbol{z}_2) - H(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2) \end{aligned} \tag{19}$$

$$\begin{aligned} & I(\boldsymbol{x}; \boldsymbol{z}_1; \boldsymbol{z}_2) \\ =& H(\boldsymbol{x}) + H(\boldsymbol{z}_1) + H(\boldsymbol{z}_2) + H(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2) \\ & - H(\boldsymbol{z}_1, \boldsymbol{z}_2) - H(\boldsymbol{x}, \boldsymbol{z}_1) - H(\boldsymbol{x}, \boldsymbol{z}_2) \end{aligned} \tag{20}$$

Based on the facts above, we have:

$$I(\boldsymbol{x}; \boldsymbol{z}_1, \boldsymbol{z}_2) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \boldsymbol{z}_2) \tag{21}$$

$$\Leftrightarrow 2H(\boldsymbol{z}_1, \boldsymbol{z}_2) + H(\boldsymbol{x}, \boldsymbol{z}_1) + H(\boldsymbol{x}, \boldsymbol{z}_2) \geq H(\boldsymbol{z}_1) + H(\boldsymbol{z}_2) + 2H(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2) \tag{22}$$

It's true because we have:

$$\begin{aligned} & H(\boldsymbol{z}_1, \boldsymbol{z}_2) + H(\boldsymbol{x}, \boldsymbol{z}_1) \\ =& H(\boldsymbol{z}_2|\boldsymbol{z}_1) + H(\boldsymbol{x}|\boldsymbol{z}_1) + 2H(\boldsymbol{z}_1) \\ \geq& H(\boldsymbol{x}, \boldsymbol{z}_2|\boldsymbol{z}_1) + 2H(\boldsymbol{z}_1) \\ =& H(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2) + H(\boldsymbol{z}_1) \end{aligned} \tag{23}$$

Similarly, the following inequality also holds true:

$$H(\boldsymbol{z}_1, \boldsymbol{z}_2) + H(\boldsymbol{x}, \boldsymbol{z}_2) \geq H(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2) + H(\boldsymbol{z}_2) \tag{24}$$

Therefore, making sum to Eq.(23) and Eq.(24), we conclude that $I(\boldsymbol{x}; \boldsymbol{z}_1, \boldsymbol{z}_2) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \boldsymbol{z}_2)$. Hence, we finish the proof of the $L = 2$ case.

When $L = k$, suppose $I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_k) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_k)$, we consider $L = k + 1$. In this case, based on the inductive assumption, we have:

$$I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{k+1}) \geq I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_k) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_k) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_{k+1}) \tag{25}$$

Hence, the case of $L = k + 1$ also holds true. Therefore, we conclude that $I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_L) \geq I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_L)$.

Now, we consider the interaction information and can obtain:

$$\begin{aligned} & I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_L) \\ =& I(\boldsymbol{z}_L, \boldsymbol{z}_{L-1}) - \sum_{i=2}^{L-1} I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_i|\boldsymbol{z}_{i-1}) - I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_1|\boldsymbol{x}) \\ \geq& \sum_{i=2}^{L-1} I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_i|\boldsymbol{z}_{i-1}) - I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_1|\boldsymbol{x}) \end{aligned} \tag{26}$$

Finally, based on Eq.(16), (17), (25), (26), we can conclude:

$$\begin{aligned} \mathbb{E}_{p(\boldsymbol{x})}[\mathcal{L}_R] =& \sum_{i=1}^{L} \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})}[\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))] \\ \geq& I(\boldsymbol{x}; \boldsymbol{z}_1, \ldots, \boldsymbol{z}_L) \\ \geq& I(\boldsymbol{x}; \boldsymbol{z}_1; \ldots; \boldsymbol{z}_L) \\ \geq& \sum_{i=2}^{L-1} I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_i|\boldsymbol{z}_{i-1}) - I(\boldsymbol{z}_L; \ldots; \boldsymbol{z}_1|\boldsymbol{x}) \end{aligned} \tag{27}$$

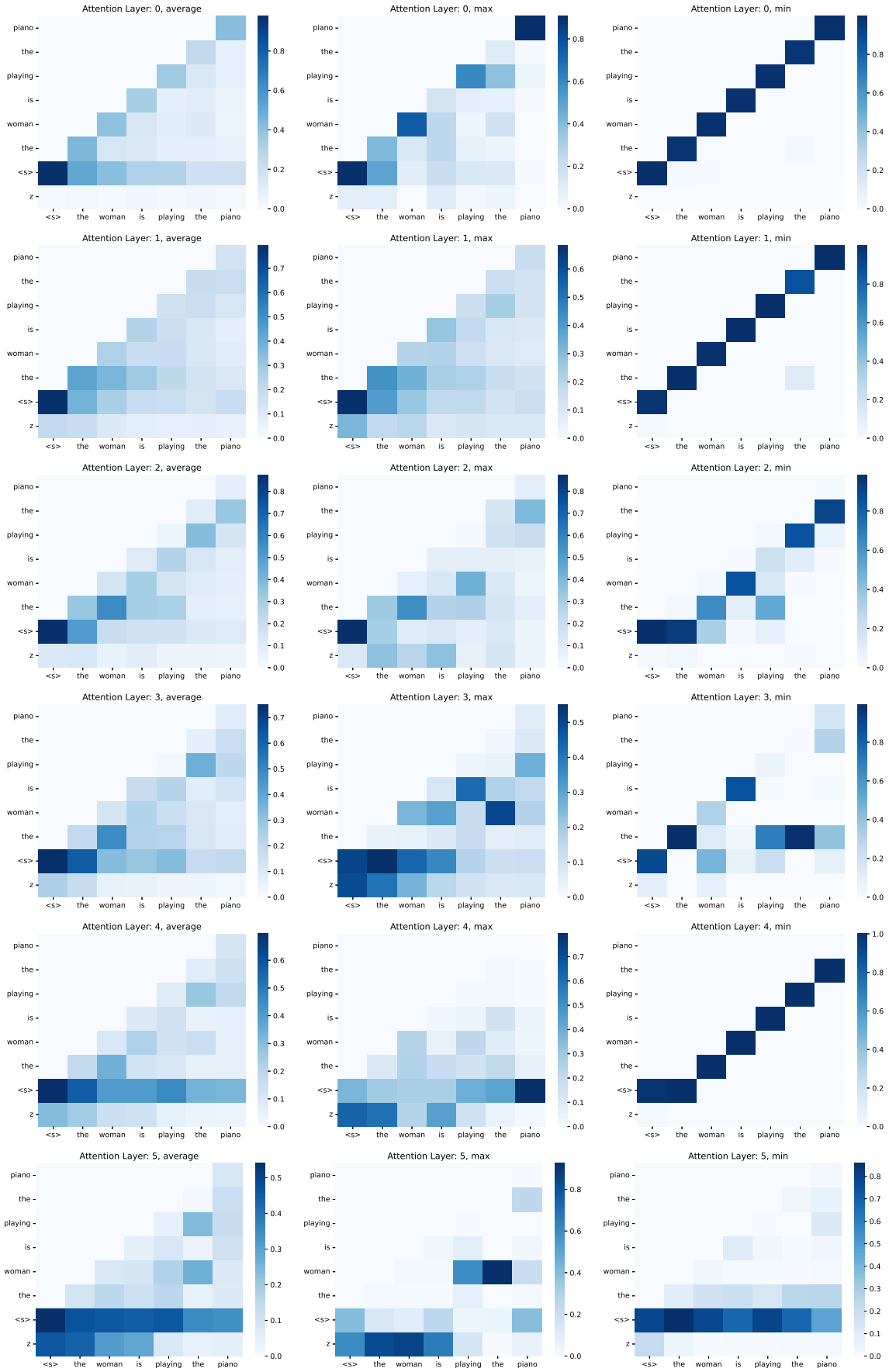

Figure 6: Attention weight of the Memory paradigm for layer 0 to layer 5. We plot three heatmaps in each layer. Average means averaging weights throught all head. Max and min means we select the head with max and min attention weight on the memory token (latent variable).We can see the memory token tends to be ignored by most heads especially in lower layers.

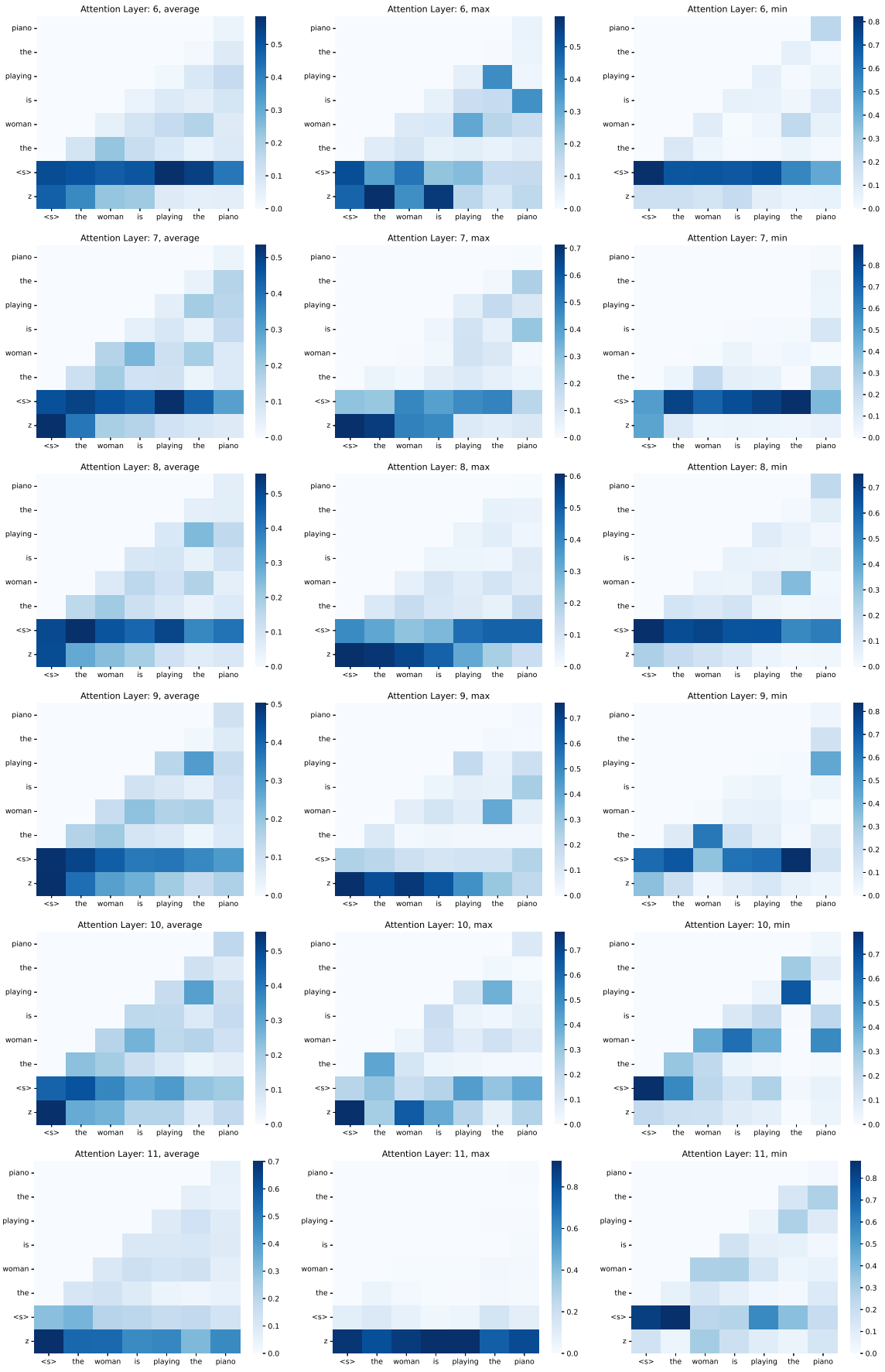

Figure 7: Attention weight of Memory paradigm for layer 6 to layer 11.