applied sciences

MDPI

Article

Informative Language Encoding by Variational Autoencoders Using Transformer

Changwon Ok [1], Geonseok Lee [2] and Kichun Lee [2,*]

[1] KT Corporation, Seongnam 13606, Korea
[2] Department of Industrial Engineering, Hanyang University, Seoul 04763, Korea
* Correspondence: skylee@hanyang.ac.kr

Abstract: In natural language processing (NLP), Transformer is widely used and has reached the state-of-the-art level in numerous NLP tasks such as language modeling, summarization, and classification. Moreover, a variational autoencoder (VAE) is an efficient generative model in representation learning, combining deep learning with statistical inference in encoded representations. However, the use of VAE in natural language processing often brings forth practical difficulties such as a posterior collapse, also known as Kullback–Leibler (KL) vanishing. To mitigate this problem, while taking advantage of the parallelization of language data processing, we propose a new language representation model as the integration of two seemingly different deep learning models, which is a Transformer model solely coupled with a variational autoencoder. We compare the proposed model with previous works, such as a VAE connected with a recurrent neural network (RNN). Our experiments with four real-life datasets show that implementation with KL annealing mitigates posterior collapses. The results also show that the proposed Transformer model outperforms RNN-based models in reconstruction and representation learning, and that the encoded representations of the proposed model are more informative than other tested models.

Keywords: natural language processing; transformer; variational autoencoder; text mining

Citation: Ok, C.; Lee, G.; Lee, K. Informative Language Encoding by Variational Autoencoders Using Transformer. *Appl. Sci.* **2022**, *12*, 7968. https://doi.org/10.3390/app12167968

Academic Editor: Valentino Santucci

Received: 15 March 2022
Accepted: 3 August 2022
Published: 9 August 2022

1. Introduction

A variational autoencoder (VAE) [1] has been applied in numerous NLP tasks, including language modeling [2] and semi-supervised text classification [3]. The most prominent component of a VAE in language modeling is the statistical use of latent representation, which aims to contain holistic and informative features in texts such as styles, topics, and semantic features. With this latent representation, samples from its prior distribution can generate diverse and exquisite sentences [2].

Due to the inherent auto-regressive nature of texts, an auto-regressive decoder such as a recurrent neural network (RNN) and a long short-term memory (LSTM) [4] are also widely used, and a few integrated models with RNN and VAE have also been proposed [2]. Previous VAE-driven language modeling approaches use encoders and decoders in the form of RNN [2] as well as the fusion of RNN and Transformer [5,6].

However, the mere use of such auto-regressive decoders, besides failing to include informative latent encodings, induces a so-called posterior collapse that makes latent representations useless [2,7]. To mitigate this problem, several techniques are proposed in the literature, such as updating inference networks and generative networks in imbalanced ways [7], refactoring loss functions [8], changing loss functions [9], and adopting Kullback–Leibler (KL) annealing [2,10].

Transformer has produced state-of-the-art outcomes, becoming a default choice in various natural language tasks, including generative language modeling [11] and discriminative language understanding [12]. Its excellent performance is due to a self-attention mechanism that captures contextual information from entire sequences. Thus, our view

is that the integration of Transformer with a variational autoencoder, which captures the entire sequence and statistical inference, will be beneficial for various natural language tasks.

To the best of our knowledge, no such models that make use of a Transformer architecture internally coupled with a VAE to build a new language modeling approach have been proposed and tested. In addition, from a computational viewpoint, Transformer has several benefits in comparison with RNNs, such as mitigating vanishing-gradient issues and the ability to handle sequential operations and parallelization [13]. Thus, building a language representation model in VAE solely with Transformer may capture the exquisite holistic features of sentences with parallel computing. Furthermore, a careful adoption of Transformer in our model may bring forth sufficient performance in various NLP tasks without a severe adaptation of the model.

To sum up, this paper makes the following contributions: (1) We provide a novel Transformer model inherently coupled with a variational autoencoder, which we call a variational autoencoder Transformer (VAE-Transformer), for language modeling; (2) We implement the VAE-Transformer model with KL annealing techniques and perform experiments involving real-life datasets with different sentence lengths. With the results, we verify that the model, which produces more informative embedding representations, is better than previous RNN-based models in reconstruction and representation learning.

2. Preliminaries

In this section, we briefly explain the models of variational autoencoders and Transformer to propose a new integrated model that combines the two.

Variational Autoencoder . A VAE model is a deep generative model using latent variables, as shown in Figure 1 [1]. VAE models assume that observable data derive from latent (hidden) variables that follow a simple distribution, usually Gaussian.

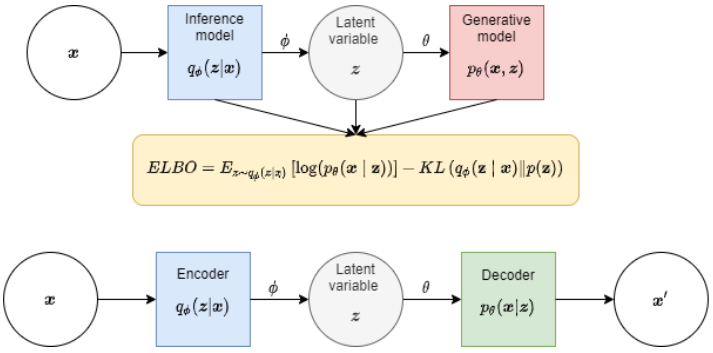

Figure 1. Variational autoencoder.

Suppose that an observed data point $\boldsymbol{x}$ is a high-dimensional random vector, and that latent variable $\boldsymbol{z}$ is in a relatively small dimensional space. The log-likelihood of data $\boldsymbol{x}$ can be expressed as follows:

$$\begin{aligned}\log p_\theta(\boldsymbol{x}) &= \log \int_z p_\theta(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z} \geq \int_z q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}d\boldsymbol{z} \\ &= \int_z q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})})d\boldsymbol{z} \\ &= E_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) := ELBO,\end{aligned} \tag{1}$$

where $p_\theta(\cdot)$ and $q_\phi(\cdot)$ are the probability density functions of $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively, and the Kullback–Leibler divergence is $KL(p||q) = \sum_x p(x) \log(\frac{p(x)}{q(x)})$. The right-hand side of the equation above is the lower bound of the log-likelihood of data $\boldsymbol{x}$, which is called an evidence lower bound (ELBO). Noticeably, by including inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and

generative model $p_\theta(\boldsymbol{x}, \boldsymbol{z})$, ELBO can be expressed as the sum of two terms by encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, as shown in Equation (2) and Figure 1. Typically, we assume that $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is Gaussian, $N(\boldsymbol{\mu}, \sigma^2\boldsymbol{I})$, with dimension k, and that $p(\boldsymbol{z})$ is also Gaussian, $N(\boldsymbol{0}, \boldsymbol{I})$, with dimension k. Under this assumption, we can rewrite ELBO as follows:

$$E_{z\sim q_\phi(z|x)}[\log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))] + \frac{1}{2}\log(|\sigma^2\boldsymbol{I}|) + \frac{k}{2} - \frac{1}{2}Tr(\sigma^2\boldsymbol{I}) - \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\mu}. \tag{2}$$

Using a neural network model of parameters ϕ and θ, we estimate the parameters by maximizing ELBO, which is equivalent to maximizing the log-likelihood. The VA model can be applied to language modeling for text generation and text embeddings. To generate a sentence, $\boldsymbol{x}$, of length T, a language model generates each token, $x_t, t \geq 1$, conditioned on the previous tokens:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t|\boldsymbol{x}_{<t}), \tag{3}$$

where x_t represents the word (or unit) token at position t, and $\boldsymbol{x}_{<t}$ represents word tokens before position t. In other words, $\boldsymbol{x}_{<t}$ means $x_1, x_2, \ldots, x_{t-1}$ for $t > 1$ and a given fixed token, <BOS>, for $t = 1$, with $p(x_1|\boldsymbol{x}_{<1}) = p(x_1)$. Using the auto-regressive characteristics of Equation (3), we express ELBO as follows:

$$E_{z\sim q_\phi(z|x)}[\sum_{t=1}^{T} \log(p_\theta(x_t|\boldsymbol{z}, \boldsymbol{x}_{<t}))] + \frac{1}{2}\log(|\sigma^2 I|) + \frac{k}{2} - \frac{1}{2}Tr(\sigma^2 I) - \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\mu} \tag{4}$$

In a variational autoencoder for language modeling, both an encoder and a decoder use auto-regressive models such as LSTM [4] and GRU [14]. As depicted in Figure 2, decoders are conditioned on latent variables designed to capture global features such as style, topic, and high-level syntactic features [2].

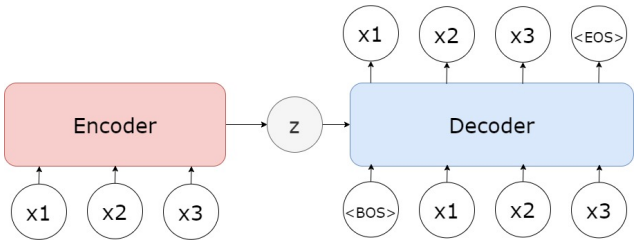

Figure 2. Illustration of a variational autoencoder for language modeling. <BOS> is the beginning of string tokens, and <EOS> is the end of string tokens

However, a variational autoencoder model using an auto-regressive decoder easily falls into local optima, which is called a posterior collapse or KL vanishing [2,7]. This phenomenon occurs because the model training implemented to maximize ELBO often finishes at an early step, with latent variables $\boldsymbol{z}$ failing to contain meaningful information about input sequences and the KL divergence term falling to zero. When reconstructing x_t with the latent variable $\boldsymbol{z}$ from such a training process, the model just relies on $\boldsymbol{x}_{<t}$, as depicted in Figure 3 by the red line [7].

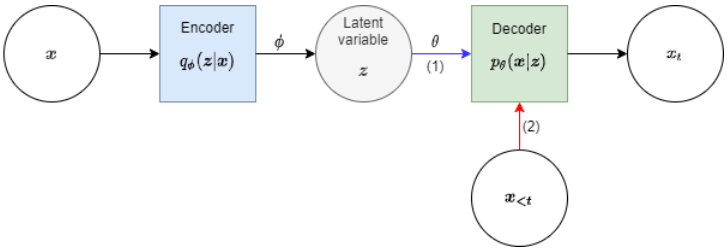

Figure 3. Illustration of a posterior collapse.

To alleviate this phenomenon, numerous techniques such as updating inference networks and generative networks in imbalanced ways [7], refactoring loss functions [8], changing loss functions [9], and KL annealing [2,10] have been proposed. Because of its fast convergence coupled with low hardware capacity, we adhere to KL annealing techniques for our experiment in this research work [2,10].

Transformer. Transformer is a sequence-to-sequence model that removes recurrence and adopts self-attention [13]. Transformer architectures have become the state of the art in various NLP tasks such as translation, summarization, and classification. Moreover, the architecture is used in various pre-trained language models such as BERT [12] and GPT [11], which have accomplished top performance in numerous NLP tasks.

Self-attention and its multi-head version are crucial in Transformer. When a sentence is injected into the Transformer architecture and then subsequently passed to the embedding layer, it is summed with positional encoding, which gives position information to the output of the embedding layer. Next, the output is projected to multiple sets of query, key, and value.

Attention includes a mapping from a query (Q), along with a key (K) and a value (V), to an output as a weighted sum of values in which the weight, also called the attention score, is the similarity between query and key. The similarity is a scaled dot product between query and key, normalized by the dimension of key, d_k, as in Equation (5), where $Q \in \mathbb{R}^{s_1 \times d_q}, K \in \mathbb{R}^{s_2 \times d_k}, V \in \mathbb{R}^{s_2 \times d_v}$, s_1 and s_2 are the lengths of the sequence in query and key (or value), respectively, and d_q, d_k, and d_v are the dimensions of query, key, and value, respectively:

$$Attention(Q, K, V) = Softmax(QK^T/\sqrt{d_k})V. \tag{5}$$

As an extension of attention, self-attention is an attention mechanism in which the query, key, and value are from same source.

In addition, the Transformer mechanism adopts a multi-head attention, a mechanism which projects query, key, and value into multiple and different spaces and applies attention functions, as shown in Figure 4. Equation (6) shows its process as a concatenation of multiple attention outputs, in which the linear projections are $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^o \in \mathbb{R}^{hd_v \times d_{model}}$, where d_{model} is the dimension of the model and h is the number of heads. Multi-head attention brings forth various views of query, key, and value. The total number of heads, which is a hyperparameter, is often set in practice by an associated performance measure.

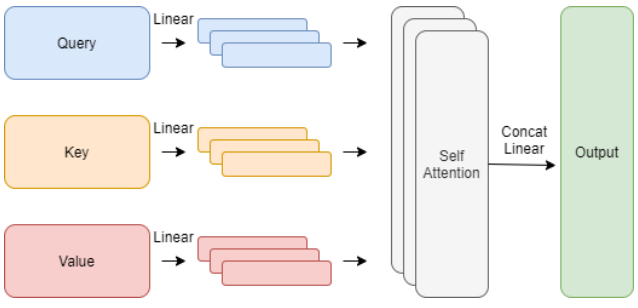

Figure 4. Illustration of multi-head attention.

$$\begin{aligned} Multi\text{-}Head(Q, K, V) &= Concat(head_{(1)}, \ldots, head_{(h)})W^o, \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V). \end{aligned} \tag{6}$$

The architecture includes an encoder composed of two sub-layers, a multi-head self-attention layer, and a feed-forward neural network layer. It also has a decoder composed of three sub-layers, a masked multi-head self-attention layer, an encoder-decoder attention layer, and a feed-forward neural network layer. The masked multi-head self-attention layer contains masks used in preventing the current token x_t from attending the next tokens.

When predicting the current token x_t, the model can see the current token x_t and previous tokens $\boldsymbol{x}_{<t}$.

Transformer is equipped with an `encoder-decoder` structure in the decoder part, as shown in Figure 5. The encoder-decoder attention layer calculates attention scores with the encoder output and the decoder input, where query is the output of the masked multi-head self-attention layer, and both key and value are the output of the encoder. We mention that the encoder-decoder structure is essentially the connection that links the encoder part to the decoder part in Transformer, which we will relate to the use of a VAE.

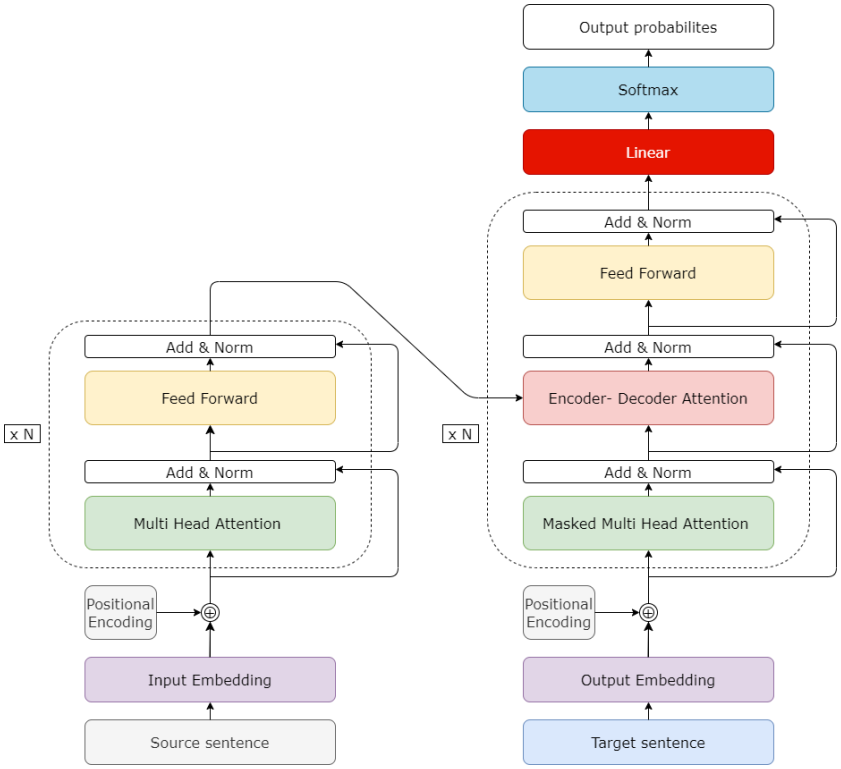

Figure 5. Illustration of the Transformer architecture.

The Transformer model is superior to existing RNN-based models in two aspects: preventing a posterior collapse and enabling parallel computing. The issue of long-term dependence occurs when recurrent models have back-propagated gradients gradually decreasing to near-zero, causing latent variable z to rarely be updated in training, which will result in a posterior collapse with useless latent variable z [7]. On the contrary, by computing all input elements equally, the Transformer model is able to capture long-term dependence and prevent posterior collapses. Preventing posterior collapses results in useful latent variables generating sentences. In addition, the Transformer model is equipped with parallel computing. Unlike Transformer, RNN-based models predict tokens in a recurrent manner so that parallel computing is impossible. Transformer, by simultaneously treating all input elements as a matrix form, enables parallel processing.

To the best of our knowledge, previous models of variational autoencoders with Transformer in language modeling use a mixed version of Transformer and RNN [5,6], in which Transformer serves as the encoder and RNNs as the decoder. With this structure, because of the auto-regressive property of RNNs, the models hardly take full advantage of Transformer. Thus, we will tightly couple VAE and Transformer, which can prevent posterior collapses and enhance parallel computing. With these ideas, we propose a VAE plugged into Transformer, which we call a VAE-Transformer model, so that the VAE can connect the output of Transformer's encoder with the input of Transformer's decoder while utilizing the ability of statistical inference via VAE in Transformer. Indeed, some Transformer-based VAE models exist in the literature. Conditional generation models with

VAE and Transformer [15,16] simply focus on predicting the next tokens of the decoder such as in a machine translation problem, where the inputs of the encoder are different from those of the decoder. The primary use of the models excludes sentence representation, which our model focuses on. Moreover, the pre-trained models coupled with VAE by [5,17] are different from our model in that they adopt pre-trained models inside. Furthermore, they fail to compare theirs with RNN-based models and only perform comparisons with other pre-trained models. Arroyo et al. [18] proposed a VAE-based Transformer model for layout generation. The previous models just used the variational autoencoder in a structural term, which did not alleviate posterior collapse. Our model's novelty is its use of VAE with Transformer in language modeling, which is applicable to conditional generation and pre-trained models. In addition, our model alleviates posterior collapse by creating a strong connection with the decoder and the encoder. We will explain our model in the following section.

3. The Proposed Model, VAE-Transformer

The proposed VAE-Transformer model combines Transformer with a variational autoencoder for both the encoder and the decoder, as shown in Figure 6. This model remains mostly the same as the original Transformer. However, the proposed model differs from the original model in that a variational autoencoder is plugged in to connect its encoder output with its encoder-decoder structure so as to apply and adjust variational effects in the decoding process. It is worth comparing the VAE-based Transformer with the original Transformer and VAE-equipped RNNs. In addition, we believe that the embedding by the VAE-based Transformer will be diverse while maintaining an effective representation of input texts, such that the proposed model may be potentially useful in further NLP tasks such as summarization, machine translation, among others.

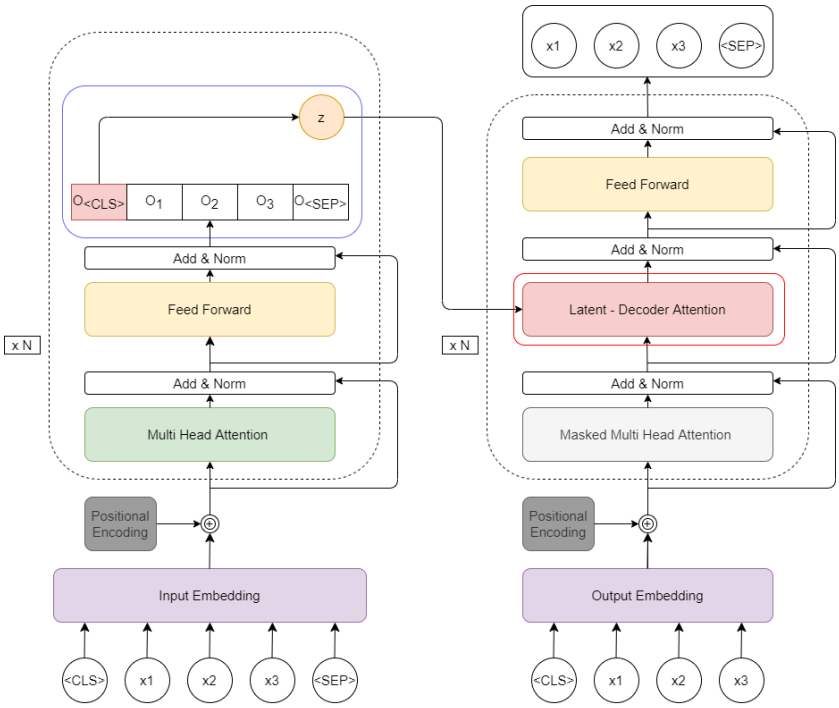

Figure 6. Illustration of a Transformer-based variational autoencoder. O is the output of the encoder layer, and the subscripts correspond with the input sentences. The latent-decoder attention layer is indicated with a red line. The latent variable z from $< CLS >$ is indicated by the blue line.

The encoder remains the same as in Transformer, except for the connection of the encoder output to the decoder by VAE. hIIn order to create a latent variable $\boldsymbol{z}$, we attach an additional neural network after the last layer of the encoder. This involves $\boldsymbol{\mu}$ and σ, which are the distribution parameters of the latent variable $\boldsymbol{z}$, which is the same as in [1] and

shown as a blue box in Figure 6. In addition, we revise the encoder-decoder attention in the Transformer decoder so that the latent variable z, instead of the output of the encoder, feeds the attention module. The rationale of this structure is that it is quite close to the encoder-decoder attention in Transformer, therefore allowing us to take advantage of the original Transformer decoder structure. Thus, by removing the previous encoder-decoder attention, we provide more attention to the latent variable and the output of the decoder. We denote this structure as a latent-decoder attention after the feed-forward network, displayed as the red box in Figure 6.

We define the abbreviations for the tokens in use: token $< CLS >$ means the beginning of a sentence, and token $< SEP >$ means the end of a sentence. Using the encoder, we generate the output of the input sequence $X = [< CLS >, x_1, x_2, \ldots, x_n]$, in which the output of $< CLS >$ represents the summary of X, which is similar to BERT [12]. Using a neural network, as the blue box shows in Figure 6, we convert the output of $< CLS >$ in the encoder into its stochastically sampled representation, z', by a Gaussian distribution, with the mean μ_z and variance σ_z of the latent variable z. Because of the indifferentiable sampling process, we use a reparametrization trick that outsources the sampling process to the additional variable, whose distribution is $N(\mathbf{0}, \boldsymbol{I})$:

$$z' = \mu_z + \sigma_z \odot \epsilon, \;\; \epsilon \sim N(\mathbf{0}, \boldsymbol{I}),$$

where $\odot$ is the element-wise product [1]. Using the decoder, we generate the embeddings, z', of the input sequence, $Y = [< CLS >, x_1, x_2, \ldots, x_n]$, and process them in the masked self-attention layer. We process the samples of latent variable z and the output of the masked self-attention layer in the latent-decoder attention layer, similar to the process performed in the encoder-decoder attention layer of the original Transformer decoder. Lastly, we predict the token x_t based on $x_{<t}$ and z. We obtain the loss of the entire output sequence by fitting it onto $[x_1, x_2, \ldots, x_n, < SEP >]$.

4. Experiments

First, we compare the proposed method with a few existing methods in language modeling. In this work, we consider three baseline models: (1) LSTM with a variational autoencoder [2], denoted by `VAE-LSTM`, which uses a variational autoencoder where both the encoder and decoder are LSTM; (2) an LSTM language model, denoted by `LSTM`, which uses an LSTM for the prediction of next tokens by previous tokens; (3) Transformer language modeling, denoted by `Transformer-Decoder`, which uses only the Transformer decoder without an encoder-decoder attention layer in the same way as [11]. We denote the proposed method as `VAE-Transformer`. We also utilize the variational autoencoder with the pre-trained model GPT2 [16] to see how a pre-trained model works in a variational autoencoder in terms of language modeling. The encoder and the decoder of this model are the Transformer encoder and pre-trained model GPT2, respectively. This model is used for conditional story generation in language modeling, and we give the same inputs to the encoder and the decoder as those used in our model. We denote this model as `VAE-GPT`. We show the parameter settings for the tested models in Table 1.

For the comparison, we used four datasets, which are publicly available and widely adopted as benchmark datasets in language modeling: `Pen Tree Bank` (PTB) [19], `Stanford Sentiment Treebank` (SST2) [20], `Twitter US Airline Sentiment` (AIR), and `WikiText2`. In short, PTB is a dataset for part-of-speech tagging in NLP, selected from Wall Street Journal. SST2 is a collection of movie reviews with two labels (positive, negative). AIR consists of tweets for US airlines with three labels (positive, neutral, negative). We, however, just use two labels (positive, negative) in the following experiments. Lastly, WikiText2 is a subset of wikipedia data, commonly used for language modeling.

To evaluate the performance of language modeling, we adopt two evaluation approaches: intrinsic and extrinsic. The intrinsic evaluation considers two aspects of the generated sentences, which are the output of the decoder. The first one is reconstruction, which refers to how well the generated sentences are reconstructed compared with the

input sentences. We represent the ability of reconstruction as a perplexity measure [2]. The definition of perplexity (PPL), shown in Equation (7), is related to a measure of how well the probability distribution of a language model predicts a text sequence as reciprocal to the averaged log probability of words. Thus, the smaller the perplexity measure for a language model, the better the model:

$$\begin{array}{ll} PPL(\boldsymbol{x}) & = P(x_1 x_2 \ldots x_N)^{-\frac{1}{N}} = \exp^{-l}, \\ where\ l & = \frac{1}{N} \log P(x_1 x_2 \ldots x_N). \end{array} \tag{7}$$

Table 1. Parameter settings for the tested models.

Model	Parameters	Value
VAE-Transformer	model dim	128
	hidden dim	512
	heads	2
	layers	2
	dropout	0.1
	activation function	GeLU
	number of parameters	12,763,706
VAE-LSTM	model dim	128
	layers	2
	encoder bidirectional	TRUE
	dropout	0.1
	number of parameters	12,526,010
VAE-GRU	model dim	128
	layers	2
	encoder bidirectional	TRUE
	dropout	0.1
	number of parameters	12,756,922
Transformer-Decoder	model dim	128
	hidden dim	512
	heads	2
	layers	2
	dropout	0.1
	activation function	GeLU
	number of parameters	8,272,186
LSTM	model dim	128
	layers	2
	dropout	0.1
	number of parameters	8,108,346
GRU	model dim	128
	layers	2
	dropout	0.1
	number of parameters	8,042,298
VAE-GPT	model dim	768
	encoder layers	2
	decoder layers	12
	dropout	0.1
	number of parameters	207,349,971

The second aspect considered by the intrinsic evaluation is how well representation learning is accomplished on the basis of a KL divergence measure since the performance of representation learning [2]. Kullback-Leibler divergence (KL) in Equation (2), being a statistical distance, is a measure of how one probability distribution is different from another reference probability distribution. In particular, if a language model produces a low reconstruction error and a high KL divergence, it means that the latent variable, z,

represents the sentence data quite well while possibly well reflecting informative features such as topics or styles from the input sentences. Furthermore, to specify how the latent variables of a language model represent data, we also include mutual information (MI) [21]. Mutual information is a measure of how much two random variables are related. We calculate the mutual information with latent variable $\boldsymbol{z}$ and input variable $\boldsymbol{x}$.

When KL divergence and mutual information are close to zero, it means that the latent variable z, unable to contain valuable information from input sentences, causes the model to degenerate as conventional language modeling does. Encoders that bring forth latent variable z with useful information from input sentences will have non-zero KL divergence and mutual information. [2] In addition, we include the extrinsic evaluation as an extension of intrinsic evaluation, especially for representation learning. If representation learning works well, the latent variable z contains an ample amount of useful information for the input sentences. To implement the usefulness of z in representation learning, we use a sentiment classification task with z for the SST2 and AIR datasets. As the classification in previous language models [11,12] proceeds with pre-training and fine-tuning after training a model for language modeling with input data, we similarly fine-tune the model for classification with the same input data, following the idea of transfer learning. For a fair comparison, we update linear classifier module parameters in the classification model, excluding VAE encoder parameters. We use latent variable $\boldsymbol{z}$ for classification, following the idea of transfer learning. Figure 7 illustrates the adopted procedure of extrinsic evaluation in classification.

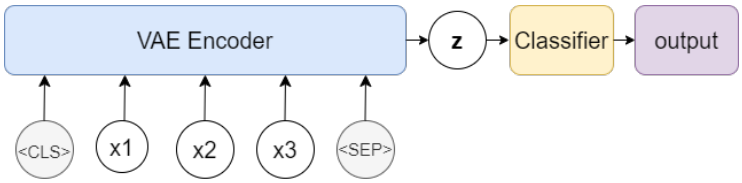

Figure 7. Procedure of extrinsic evaluation in classification.

For pre-processing input texts in the experiments, we use a sentence-piece tokenizer with a vocabulary size of 30,522, similar to the procedure used for BERT [22]. For `VAE-GPT`, however, we use a byte pair encoding tokenizer that is used in GPT2. We use a fixed sequence length (including $< BOS >$ and $< EOS >$ tokens). When an input sentence is larger than the fixed length, it is truncated. When it is less than the length, it is zero-padded. We use a fixed sequence length of 64 for the PTB, SST2, and AIR datasets and 256 for the Wikitext2 dataset.

The parameters of the tested models are shown in Table 1. For fair testing, we make the total number of parameters for the VAE-Transformer and VAE-LSTM models as close as possible. We also make those for the Transformer-Decoder and LSTM as close as possible. We notice that the parameters of the Transformer-Decoder and LSTM without VAEs are smaller than those of the VAE-Transformer and VAE-LSTM with VAEs. As it is a pre-trained model, VAE-GPT is the largest model of all, and a direct model comparison is fair. However, we can check the effect of a pre-trained model in the variational autoencoder.

The model dimension of the encoder and decoder of the Transformer model is 128, and its hidden dimension is 512. Furthermore, we use two multi-head attention mechanisms, the GeLU activation function from Ref. [23] in each layer, and the dimension of 64 for the latent variables. We set the encoder of the VAE-GPT model to be the same as the VAE-Transformer encoder and the decoder of the VAE-GPT model the same as in [16].

We set the encoder of the LSTM model to be bidirectional and the decoder to be unidirectional. We set the hidden dimension as 128, and 2 hidden layers for both the encoder and the decoder. The hidden dimension of the LSTM language modeling is set at 128, and the two hidden layers are set to be the same as the VAE-LSTM. Moreover, we use the same settings as that used with the VAE-Transformer for the Transformer-Decoder.

We optimize the models by the AdamW optimizer [24] with cosine annealing [25] for a varying learning rate. We use an initial learning rate of 1×10^{-3} for the PTB and Wikitext2 datasets and 1×10^{-4} for the SST2 and AIR datasets. The final learning rate of 1×10^{-7} is used, which restarts every 10 epochs for all datasets. We set the momentum parameters at 0.9 and 0.999, and the weight decay as 0.1. The batch size is 32, and our model is trained in 50 epochs.

To prevent KL vanishing, we use linear annealing [2] and cyclical annealing [10]. With the annealing techniques, the objectives of the model (ELBO) are rewritten as in Equation (8). Notice that $\alpha(t)$ represents a learning rate that varies with iteration t.

$$ELBO = E_{z\ q_\phi(z|x)}[\log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))] - \alpha(t)\, KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{8}$$

For linear annealing, $\alpha(t)$ starts at 0 and reaches 1 at a certain step, after which we set $\alpha(t)$ at 1. For cyclic annealing, $\alpha(t)$ is the same as in linear annealing, except that after reaching 1, it resets to 0 and repeats this several times, as shown in Equation (9). Figure 8 shows the comparison between linear annealing and cyclic annealing scheduling according to iteration t.

$$\begin{aligned} \alpha(t) &= \begin{cases} f(\tau), & \tau \leq R, \\ 1, & \tau > R, \end{cases} \\ \tau &= \frac{\mathrm{mod}\ (t, \lceil T/M \rceil)}{T/M}, \end{aligned} \tag{9}$$

where T is the total number of training steps, f is a monotonically increasing function, M is the number of cycles, and R is the proportion used to increase $\alpha(t)$ within a cycle. We set R at 0.5 and M at 5 for 50 training epochs, in the same way as [10].

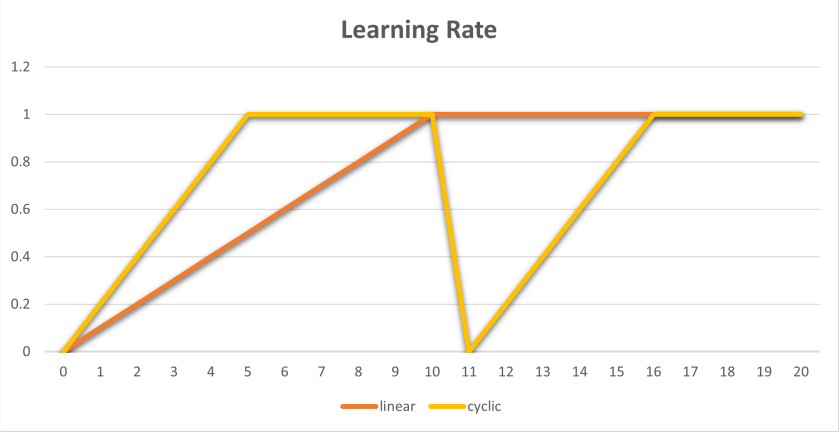

Figure 8. Comparison between linear annealing and cyclic annealing scheduling.

4.1. Results

4.1.1. Intrinsic Evaluation

As described in the previous section, we compared our model, the `VAE-Transformer`, with `VAE-LSTM`, `LSTM LM`, `Transformer Decoder`, and `VAE-GPT` using intrinsic evaluation.

Table 2 shows that the `VAE-Transformer` outperforms the LSTM-based variational autoencoder, LSTM language modeling, and Transformer Decoder in terms of reconstruction and representation learning. It can be observed that the proposed model, the `VAE-Transformer`, is better than `VAE-LSTM` regardless of the KL annealing strategy. Table 2 also shows that using a pre-trained model with VAE improves the performance of a language model. This is because of the knowledge transfer and the size of the pre-trained model. To illustrate the training performance, we show the learning curves of cross entropy and KL divergence according to epochs for `VAE-Transformer` and `VAE-LSTM` in the training session of PTB, as shown in Figure 9. In terms of cross entropy, the `VAE-Transformer` is much lower than the `LSTM`. Moreover, it converges faster than `LSTM`. In each of the KL

annealing strategies, the KL divergence of the `VAE-Transformer` was larger than that of VAE-LSTM.

Table 2. Language modeling evaluation in terms of perplexity (PPL), KL divergence (KL), and mutual information (MI). *Linear* means KL annealing with a linear function, and *Cycle* means KL annealing with a cyclic function. Bold-faced numbers and italic ones represent the best and the second best, respectively.

	PTB			**Wikitext2**			**SST2**			**AIR**		
Model	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**
VAE+ Transformer+*Cycle*	*75.65*	**4.40**	**2.45**	*121.71*	*4.27*	*1.96*	411.40	*0.21*	*0.11*	*83.70*	*0.70*	**0.62**
VAE+Transformer+*Linear*	*82.01*	*2.50*	*1.54*	*120.16*	**5.29**	**2.26**	*408.62*	0.18	0.08	85.92	0.01	0.01
VAE+Transformer	91.52	0.00	0.00	128.25	0.16	0.09	413.77	0.06	0.02	88.65	0.00	0.00
Transformer Decoder	118.69			170.42			456.60			100.90		
VAE+LSTM+*Cycle*	88.64	0.90	0.81	133.81	0.55	0.38	659.85	0.00	0.00	142.72	0.01	0.01
VAE+LSTM+*Linear*	92.08	0.01	0.01	134.61	0.01	0.01	629.05	0.00	0.00	137.53	0.00	0.00
VAE+LSTM	92.04	0.00	0.00	132.23	0.03	0.02	656.12	0.00	0.00	147.00	0.00	0.00
LSTM LM	102.44			151.86			646.63			152.25		
VAE+GPT	**54.23**	0.01	0.00	**79.10**	0.02	0.01	**358.90**	**1.57**	**0.74**	**83.44**	**0.86**	*0.52*

The experimental results indicate that the latent variables of our model contain more useful information from input sentences than the `VAE-LSTM`. Furthermore, in comparison with `LSTM-LM` and `Transformer Decoder` in reconstruction measurement (PPL), the proposed model predicts the next word better than the compared models. This indicates the informative representation z of the proposed model. In PTB and Wikitext2, `VAE-GPT` has the best results for reconstruction measurement (PPL), while its KL divergence and MI are low. This implies that posterior collapse happens naturally in those cases because of the complexity and power of the used pre-trained model. Taking advantage of pre-trained knowledge, the pre-trained model is able to reproduce outputs without relying on latent variables. On the contrary, the results of SST2 and AIR, which consist of a small amount of data, are quite different, and the decoder of the variational autoencoder has to rely on latent variables in reproducing the sentences.

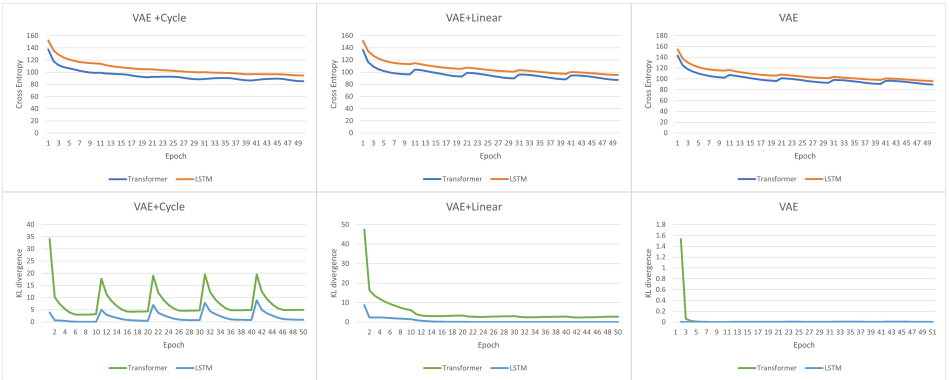

Figure 9. Learning curves in training the PTB dataset.

4.1.2. Extrinsic Evaluation

In the SST2 dataset, the `VAE-Transformer` using linear annealing yielded the best result, but in order to achieve consistency with the results of [2,10], we used cyclic annealing models for extrinsic evaluation. Similarly, in pre-trained models, we fine-tuned just the encoder part of the variational autoencoders in the pre-trained language model and chose

the best model using the validation data. The model was optimized by the AdamW optimizer [24] with cosine annealing [25]. We used an initial learning rate of 1×10^{-4}, with a final learning rate of 1×10^{-7} that restarted every 10 epochs for all datasets. We set the momentum parameters at 0.9 and 0.999 and the weight decay at 0.1. The batch size was 32, and we trained the classification model for extrinsic evaluation in 20 epochs. The result is shown in Table 3. For all the datasets, it can be seen that the proposed model shows better performance than `VAE-LSTM` in cross entropy. This result demonstrates that the latent variable of the proposed model is more informative and fit for classification tasks than that of `VAE-LSTM`. In addition, we can observe that `VAE-GPT` has the largest cross entropy, which implies that with the powerful and complex decoder, the latent variables have low sentence representation capability because of posterior collapse.

With intrinsic evaluation and extrinsic evaluation, our model surpassed the other tested models. Overall, the experimental results show that the Transformer-based models are better than RNN-based model in reconstruction and representation learning, and better at avoiding posterior collapse than the RNN-based models.

Table 3. The results of classification of the SST2 and AIR datasets. BCE represents binary cross entropy, and Acc represents accuracy.

Model	SST2						AIR					
	Train		Validation		Test		Train		Validation		Test	
	BCE	Acc	BCE	Acc	BCE	Acc	BCE	Acc	BCE	Acc	BCE	Acc
VAE+ Transformer+Cycle	0.630	0.689	0.641	0.661	**0.618**	**0.700**	0.492	0.793	0.459	0.797	**0.483**	0.800
VAE+LSTM+Cycle	0.638	0.682	0.649	0.652	0.631	0.694	0.547	0.794	0.543	0.800	0.542	**0.801**
VAE+GPT	0.634	0.670	0.650	0.646	0.633	0.677	0.522	0.787	0.517	0.789	0.525	0.798

4.1.3. Ablation Study

We also checked the consistency of the proposed model by performing experiments with different latent dimensions in intrinsic evaluation. We show the results in Table 4 for latent dimension 32. The results for latent dimension 32 are similar with the results for latent dimension 64 in language modeling and representation learning. Because of the size of the model, practical scores are different. However, both results show that the variational autoencoder for Transformer reached the best results in every dataset. These results show that our proposed model is superior to the other models in language modeling and representation learning.

Table 4. Language modeling evaluation. These models have a latent variable whose dimension is 32. Bold-faced and italic numbers represent the best and the second best, respectively.

	PTB			Wikitext2			SST2			AIR		
Model	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**
VAE+ Transformer+Cycle	**75.17**	**4.78**	**2.58**	*125.78*	*3.67*	*1.73*	**301.83**	**1.02**	**0.90**	**71.40**	**2.64**	**2.14**
VAE+Transformer+Linear	*81.49*	*2.70*	*1.65*	**122.17**	**5.48**	**2.33**	*312.05*	*0.01*	*0.01*	*77.95*	*0.33*	*0.30*
VAE+Transformer	92.23	0.00	0.00	128.46	1.28	0.68	321.14	0.01	0.00	80.23	0.00	0.00
Transformer Decoder	96.79			160.97			358.63			79.08		
VAE+LSTM+Cycle	88.31	0.91	0.82	131.67	1.38	0.82	504.19	0.00	0.00	118.02	0.03	0.03
VAE+LSTM+Linear	90.81	0.01	0.00	136.60	0.00	0.00	503.56	0.00	0.00	115.16	0.00	0.00
VAE+LSTM	92.79	0.00	0.00	134.02	0.00	0.00	547.01	0.00	0.00	121.89	0.00	0.00
LSTM LM	93.30			141.90			486.67			123.43		

Moreover, we can see the effect of LSTM in the model by replacing it with GRU. We show the results in Tables 5 and 6, with the proposed model evaluated by `VAE-GRU`.

Table 5. Language modeling evaluation. These models have a latent variable whose dimension is 32. Bold-faced and italic numbers represent the best and the second best, respectively.

	PTB			Wikitext2			SST2			AIR		
Model	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**	**PPL**	**KL**	**MI**
VAE+ Transformer+Cycle	*75.17*	*4.78*	*2.58*	125.78	3.67	1.73	**301.83**	**1.02**	**0.90**	**71.40**	**2.64**	**2.14**
VAE+Transformer+Linear	81.49	2.70	1.65	122.17	5.48	2.33	*312.05*	*0.01*	*0.01*	*77.95*	*0.33*	*0.30*
VAE+Transformer	92.23	0.00	0.00	128.46	1.28	0.68	321.14	0.01	0.00	80.23	0.00	0.00
Transformer Decoder	96.79			160.97			358.63			79.08		
VAE+LSTM+Cycle	88.31	0.91	0.82	131.67	1.38	0.82	504.19	0.00	0.00	118.02	0.03	0.03
VAE+LSTM+Linear	90.81	0.01	0.00	136.60	0.00	0.00	503.56	0.00	0.00	115.16	0.00	0.00
VAE+LSTM	92.79	0.00	0.00	134.02	0.00	0.00	547.01	0.00	0.00	121.89	0.00	0.00
LSTM LM	93.30			141.90			486.67			123.43		
VAE+GRU+Cycle	**69.88**	**5.99**	**2.94**	**110.96**	**9.34**	**2.88**	381.45	0.09	0.08	99.00	0.29	0.24
VAE+GRU+Linear	76.33	3.57	2.07	*110.15*	*9.08*	*2.87*	394.33	0.00	0.00	102.11	0.03	0.02
VAE+GRU	86.74	0.70	0.46	123.66	0.68	0.23	376.84	0.00	0.00	95.96	0.00	0.00
GRU LM	89.92			123.56			376.72			96.60		

Table 6. The results of the classification of the SST2 and AIR datasets. BCE represents binary cross entropy, and Acc represents accuracy. These models have a latent variable whose dimension is 64.

Model	SST2						AIR					
	Train		**Validation**		**Test**		**Train**		**Validation**		**Test**	
	BCE	**Acc**	**BCE**	**Acc**	**BCE**	**Acc**	**BCE**	**Acc**	**BCE**	**Acc**	**BCE**	**Acc**
VAE+ Transformer+Cycle	0.630	0.689	0.641	0.661	**0.618**	**0.700**	0.492	0.793	0.459	0.797	**0.483**	0.800
VAE+LSTM+Cycle	0.638	0.682	0.649	0.652	0.631	0.694	0.547	0.794	0.543	0.800	0.542	**0.801**
VAE+GRU+Cycle	0.630	0.689	0.643	0.661	0.619	0.700	0.535	0.794	0.528	0.800	0.590	0.801

The results show that VAE without annealing produces a score of almost zero in KL divergence and Mutual information. This is because of posterior collapse, after which the latent variable becomes a useless representation. It can be observed that the GRU-based model achieved the best performance in the PTB and Wikitext2 datasets, while the Transformer-based model was the best in the SST2 and AIR datasets in the language modeling task.

As for the model's computation time, we measured the time required to calculate forward passes in training the data, which is shown in Table 7. We conducted the experiments on both PTB and WikiText2, whose volumes were substantially large. To check the parallelization of the model, we sized up the model in terms of the parameters. As shown in Table 7, when the model's dimension is 128, the differences between `VAE-Transformer`, `VAE-LSTM`, and `VAE-GRU` are indiscernible even though `VAE-GRU` is the fastest. However, when the dimension is 1024, the differences between `VAE-Transformer`, `VAE-LSTM`, and `VAE-GRU` become discernible. Even though `VAE-Transformer` has the largest number of parameters, as shown in Table 8, `VAE-Transformer` is the fastest. It shows that Transformer-based variational autoencoders possess great parallelization as compared to RNN-based variational autoencoders.

Table 7. The results for the language modeling computation time (in seconds) in the PTB and Wikitext2 datasets.

Data	PTB		Wiktext2	
Model's dimension	128	1024	128	1024
VAE+Transformer	14.51	**85.84**	23.12	**142.48**
VAE+ LSTM	12.64	137.36	19.05	227.65
VAE+GRU	**11.29**	109.78	**16.44**	182.79
Transformer	11.39	63.09	19.84	106.70
LSTM	9.89	74.37	15.14	119.79
GRU	9.44	60.44	14.81	104.28

Table 8. The number of parameters in Table 7.

	Model's Dimension	Number of Parameters
VAE+Transformer	128	12,763,706
	1024	118,395,834
VAE+LSTM	128	12,756,922
	1024	87,515,578
VAE+GRU	128	12,526,010
	1024	75,575,738
Transformer	128	8,272,186
	1024	71,461,690
LSTM	128	8,108,346
	1024	48,315,450
GRU	128	8,042,298
	1024	45,034,554

In the classification task, the GRU-based model achieved the best results in terms of computational time. One needs to be wary of concluding that the proposed model is inferior to the GRU-based models. Naturally, the size of a GRU-based model is smaller than Transformer- and LSTM-based models. For example, the GRU-based model, which converges faster than the other models, has 12,526,010 parameters, while the Transformer-based model and the LSTM-based model have 12,763,706 and 12,756,922, respectively. We recommend that future research should study the effect of LSTM variants with a similar size and compare them to adequately fit datasets.

5. Conclusions

This paper introduced the use of a Transformer-based variational autoencoder for sentences. We compared the model with an RNN-based variational autoencoder, Transformer decoder, and LSTM. Our model trained input datasets faster than the RNN-based model, and our model was better than the RNN-based model in terms of reconstruction and representation learning. Furthermore, we used our model in a classification task. The results show that the latent variables of our model exceeded those of the RNN-based model, which implies the ability of the model to learn the holistic features of input sentences. Moreover, we also infer that with a strong and complex decoder such as the pre-trained model GPT, posterior collapse might occur. In future work, we hope to combine our model of state-of-the-art pre-trained word embeddings such as GPT [11], BERT [12], and BART [26] with techniques that reduce posterior collapse. In this work, we used the traditional loss function and its KL annealing-equipped version in a variational autoencoder to address

the issue of posterior collapse. We also aimed to compare various techniques, including other loss functions, to alleviate posterior collapse in further studies.

Author Contributions: Investigation, C.O.; Resources, G.L.; Supervision, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2020R1F1A1076278). This work was also supported by 'Human Resources Program in Energy Technology' of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea (No. 20204010600090).

Conflicts of Interest: The authors declare that there are no conflict of interest regarding publication of this paper.

References

1. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
2. Bowman, S.R.; Vilnis, L.; Vinyals, O.; Dai, A.M.; Jozefowicz, R.; Bengio, S. Generating sentences from a continuous space. *arXiv* **2015**, arXiv:1511.06349.
3. Xu, W.; Sun, H.; Deng, C.; Tan, Y. Variational autoencoder for semi-supervised text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; p. 31.
4. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
5. Liu, D.; Liu, G. A transformer-based variational autoencoder for sentence generation. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–7.
6. Duan, Y.; Xu, C.; Pei, J.; Han, J.; Li, C. Pre-train and plug-in: Flexible conditional text generation with variational auto-encoders. *arXiv* **2019**, arXiv:1911.03882.
7. He, J.; Spokoyny, D.; Neubig, G.; Berg-Kirkpatrick, T. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv* **2019**, arXiv:1901.05534.
8. Zhao, S.; Song, J.; Ermon, S. Infovae: Information maximizing variational autoencoders. *arXiv* **2017**, arXiv:1706.02262.
9. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. *Int. Conf. Mach. Learn.* **2017**, *70*, 214–223.
10. Fu, H.; Li, C.; Liu, X.; Gao, J.; Celikyilmaz, A.; Carin, L. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv* **2019**, arXiv:1903.10145.
11. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving language understanding by generative pre-training. *Preprint* **2018**, *in press*.
12. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
14. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
15. Wang, T.; Wan, X. T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
16. Fang, L.; Zeng, T.; Liu, C.; Bo, L.; Dong, W.; Chen, C. Transformer-based Conditional Variational Autoencoder for Controllable Story Generation. *arXiv* **2021**, arXiv:2101.00828.
17. Park, S.; Lee, J. Finetuning Pretrained Transformers into Variational Autoencoders. *arXiv* **2021**, arXiv:2108.02446.
18. Arroyo, D.M.; Postels, J.; Tombari, F. Variational transformer networks for layout generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13642–13652.
19. Marcus, M.; Santorini, B.; Marcinkiewicz, M. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist.* **1993**, *19*, 313–330.
20. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
21. Hoffman, M.D.; Johnson, M.J. Elbo surgery: Yet another way to carve up the variational evidence lower bound. In Proceedings of the Workshop in Advances in Approximate Bayesian Inference, NIPS, Barcelona, Spain, 9 December 2016; Volume 1.
22. Taku, K.; John, R. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
23. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv* **2016**, arXiv:1606.08415.
24. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.

25. Loshchilov, I.; Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
26. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.