VOLTA: Diverse and Controllable Question Generation with Variational-Mutual-Information-Maximizing VAE

Anonymous ACL submission

Abstract

Most recent natural language generation models only focus on the quality of the generated text, which is usually measured against a set of reference sentences. This causes the models to generate similar sentences given the same context and thus leads to low diversity in the generated content. In this paper, we propose a model named **VOLTA** that leverages the Variational Autoencoder framework to improve the diversity of large-scale language models. Unlike the prior attempts, we use a shared GPT-2 backbone network for both the encoder and the decoder because it has proved to be effective in both natural language understanding and generation. In addition, we propose to add latent codes that originated from InfoGAN to enable input-independent controllability. Our model architecture can be used for any typical language generation tasks, but we test it on the question-answer pair generation task as it has series of well-established evaluation metrics. Experimental results show that our model can significantly improve the generative diversity over previous models.

1 Introduction

Natural language generation (NLG) is an important aspect of natural language processing (NLP), including tasks such as question generation (Xiao et al., 2020a), dialog generation (Liu et al., 2020) and machine translation (Edunov et al., 2018), etc. A series of pre-trained language models (PLMs) based on Transformers (Radford et al., 2019; Devlin et al., 2019) were introduced for the NLG tasks, such as GPT (Radford et al., 2019).

Although many PLMs achieved good performance on the NLG tasks, the top generated sentences are usually very similar to each other. The cause is that regular PLMs do not have a dedicated structure to adjust the embeddings of the input and, in turn, to change the generated text. Variational Autoencoders (VAE) (Kingma and Welling, 2014) provides a framework where, with the addition of low-dimensional latent variables, the model can encode input into an organized latent space, which can then be used to dictate the decoding process. By perturbing the latent variables, the generated sentences can divert away from the few best sentences, which corresponds to improved diversity.

Context	Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary . Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto , a Marian place of prayer and reflection. . . . . . .
Q1	What type of statue is on the main building?
A1	golden statue of the Virgin Mary
Q2	What is the name of the copper statue on the main building?
A2	a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes".
Q3	What is next to the main building?
A3	Grotto

Table 1: An example of diverse QAG by VOLTA.

The challenge of introducing Transformer models into the VAE framework lies in that they are highly parallelized models where a sequence of contextualized token embeddings are passed through the model simultaneously. In this scenario, it is difficult to add a bottleneck layer of latent variables to the Transformer model itself. Optimus (Li et al., 2020) used BERT (Devlin et al., 2019) as the encoder and GPT-2 (Radford et al., 2019) as the decoder, and proposed two ways to connect latent variables to the two Transformer models: "embedding" and "memory". It is the first large-scale PLM built under the VAE framework and achieved the state-of-the-art performance on several NLG tasks, such as dialog response generation, stylized response generation, label-conditional text generation, etc. Our model differs from Optimus in that we do not use BERT as the VAE encoder. Instead, we share a GPT-2 backbone for both the encoder and the decoder. The reason why this is possible is that GPT-2 has proved to be effective in both natural language understanding and natural language generation (Radford et al., 2018, 2019; Brown et al., 2020). By doing this, we can vastly decrease the model size by half. In addition, it also simplifies the tokenization process.

Besides text generation diversity, VAE also provides a certain degree of controllability. For instance, one can interpolate between two latent variables to generate a series of different text. However, the latent variables are largely dependent on the input context. To introduce another input-independent method to control the generation process, we draw inspiration from InfoGAN (Chen et al., 2016). It proposed to add latent codes to the input noise when training a GAN model (Goodfellow et al., 2020). By optimizing a novel Variational Mutual Information Maximization objective, the generator can automatically discover different types of semantic features via the latent codes, and the generated content can be controlled by the latent codes. For the MNIST dataset (LeCun et al., 1998), the discrete latent codes can vary the type of the generated digits and the continuous latent codes can adjust their rotation and width. Our model does not follow the GAN framework but leverages latent codes to inject controllability into the PLMs. To the best of our knowledge, our work is the first one to add latent codes to PLMs. Because our model follows the VAE framework and uses the Variational Mutual Information Maximization objective from InfoGAN, we name it **VOLTA** (**V**ariati**O**nal-Mutua**L**-Informa**T**ion-Maximizing V**A**E).

Our model can be used for any typical NLG tasks, but we apply it to the question-answer pair generation task (QAG) because it has a variety of well-established metrics for evaluating the quality and diversity of the generated content. QAG aims to generate a pair of a question and an answer based on the a provided context. The answer is a text span in the context, while the question should be closely related to the answer. A QAG model can be used to augment a question-answering dataset by generating new question-answer pairs, enabling semi-supervised learning for downstream question-answering models.

The main contributions of this paper are:

- VOLTA is the first to introduce a large-scale PLM under the VAE framework for the question-answer pair generation task; in addition, it reduces the model size by half compared to Optimus (Li et al., 2020) with the shared GPT-2 backbone;

- We are the first to propose adding latent codes to PLMs for input-independent controllability; this is also the first work that combines latent codes with VAE latent variables in the field of NLP;

- Comprehensive experimental results on the question-answer pair generation task show the effectiveness of our model in improving diversity and controllability.

2 Related Work

Many Transformer-based PLM models with a large variety of configurations were introduced in recent years, including BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020), T5 (Raffel et al., 2020), etc. But most of them do not focus on the diversity or the controllability of the generative process.

Variational Autoencoders (VAE) (Kingma and Welling, 2014) differ from Autoencoders (AEs) (Hinton and Salakhutdinov, 2006) in the addition of the low-dimensional latent variables. It was originally used in Computer Vision and then adapted to NLP. Early attempts (Rezende et al., 2014; Kingma et al., 2016; Bahuleyan et al., 2018) used LSTM (Hochreiter and Schmidhuber, 1997) as the encoder and the deocder, such as Info-HCVAE (Lee et al., 2020). They were mostly successful in achieving guided sentence generation but also inherit the limitations of LSTM. Recent works combined large PLMs with VAE and generated better results. For example, Optimus (Li et al., 2020) used BERT as the encoder and GPT-2 as the decoder. Optimus outperforms LSTM-based models in VAE language modeling.

To achieve controllable language generation, some methods add special prompt tokens or control phrases to control the generated sentences. For example, SimpleTOD (Hosseini-Asl et al., 2020) adds different prompt tokens to make GPT-2 generate different dialogue responses. Similar methods include CTRL (Keskar et al., 2019), Soloist (Peng et al., 2021), CGRG (Wu et al., 2021), and MEGATRON-CNTRL (Xu et al., 2020). Dathathri et al. (2020) proposed the Plug and Play Language Model (PPLM) to guide language generation by plugging simple attribute classifiers into existing language models and it does not need re-training the models. These methods require little to none modification to the Transformer models because they mainly rely on changing the input sequences and the output targets.

InfoGAN (Chen et al., 2016) was first introduced to discover latent modalities in the MNIST dataset (LeCun et al., 1998) in an unsupervised manner. The generated images can be controlled by latent codes after training InfoGAN with the Variational Mutual Information Maximization objective. There are also attempts to combine InfoGAN with VAE to create diverse and controllable generative models, such as VAE-Info-cGAN (Xiao et al., 2020b) and InfoVAEGAN (Ye and Bors, 2021). But neither of then are for NLP. There are also works that apply mutual information to VAE, such as Info-VAE (Zhao et al., 2019) and InfoMax-VAE (Lotfi-Rezaabad and Vishwanath, 2020). However, they maximize mutual information to solve the latent variable collapse problem (Chen et al., 2017) and there is no addition of the desired latent codes. To the best of our knowledge, our model is the first to combine large PLMs with VAE and InfoGAN.

3 Our Method

We design our model to enable diverse and controllable language generation using the Variational Autoencoder framework (Kingma and Welling, 2014) and latent codes from InfoGAN (Chen et al., 2016). The VAE framework produces latent variables that encode the input information. By perturbing the latent variables, one can change the decoded content slightly and achieve more diversity. Unlike VAE latent variables, InfoGAN latent codes is input-independent. That is, their values are not determined by the input but by human. This provides another way to control the generated sequence. The overview of our model is shown in Figure 2.

3.1 Preliminaries

The question-answer pair generation task aims to generate a pair of question $\boldsymbol{x}_{qtn}$ and answer $\boldsymbol{x}_{ans}$ based on the given context $\boldsymbol{x}_{ctx}$. The context $\boldsymbol{x}_{ctx} = (x_1, \ldots, x_m)$ and the question $\boldsymbol{x}_{qtn} = (x'_1, \ldots, x'_n)$ are both sequences of tokens, while the answer $\boldsymbol{x}_{ans} = (start, end) \in \mathbb{Z}^2$ is a pair of integer indices, specifying the start and the end of the answer span in the context. That is, the answer sequence $(x_{start}, \ldots, x_{end})$ can be found by looking into the context sequence $\boldsymbol{x}_{ctx} = (x_1, \ldots, x_m)$ based on the answer span $\boldsymbol{x}_{ans}$. The goal is to find a model f that can generate a pair of question and answer using the known context: $f(\boldsymbol{x}_{ctx}) \rightarrow (\boldsymbol{x}_{qtn}, \boldsymbol{x}_{ans})$. We use $\boldsymbol{x} = [\boldsymbol{x}_{ctx}, \boldsymbol{x}_{qtn}, \boldsymbol{x}_{ans}]$ to denote the input containing context, question and answer.

3.2 Latent Variables

Similar to Optimus (Li et al., 2020), our model follows the Variational Autoencoder (VAE) framework (Rezende et al., 2014; Kingma et al., 2016; Bahuleyan et al., 2018), where the encoder f_θ and the decoder f_ϕ are both Transformer models. Both our model and Optimus use GPT-2 as the decoder f_ϕ but the difference is that Optimus uses a separate BERT (Devlin et al., 2019) model as the encoder f_θ while our model shares a GPT-2 (Radford et al., 2019) backbone network for both the encoder and decoder.

The encoder encodes the question and the answer into two different sets of latent variables. We use a set of continuous latent variables to capture the question information while we model answers with a set of discrete latent variables:

$$\begin{aligned} \boldsymbol{\mu}, \boldsymbol{\sigma}^2 &= \mathrm{MLP}(f_\theta(\boldsymbol{x}_{qtn})) \\ \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_p &= \mathrm{MLP}(f_\theta(\boldsymbol{x}_{ctx}, \boldsymbol{x}_{qtn}, \boldsymbol{x}_{ans})) \\ \boldsymbol{z}_q &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \\ \boldsymbol{z}_a &\sim [\mathrm{Cat}(\boldsymbol{\pi}_1), \ldots, \mathrm{Cat}(\boldsymbol{\pi}_p)] \, , \end{aligned} \tag{1}$$

where $\mathrm{MLP}(\cdot)$ is a fully-connected layer and each instance is distinct and has a different set of learnable parameters; $\mathcal{N}(\cdot)$ is the multivariate Gaussian distribution and its parameters are $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$; $\mathrm{Cat}(\cdot)$ is the categorical distribution whose parameters $\boldsymbol{\pi}$ represent the event probabilities of k categories, and the encoder produces p independent such latent variables. To allow gradient to be back-propagated through the latent variables, the Gaussian distribution reparametrization trick (Wolpe and de Waal, 2019) is used for $\boldsymbol{z}_q$; for $\boldsymbol{z}_a$, we use Gumbel-Softmax (Maddison et al., 2017; Jang et al., 2017) to reparameterize the categorical distribution.

Since the Kullback–Leibler divergence between the learned distribution and the prior distribution cannot be optimized directly, we use the Evidence

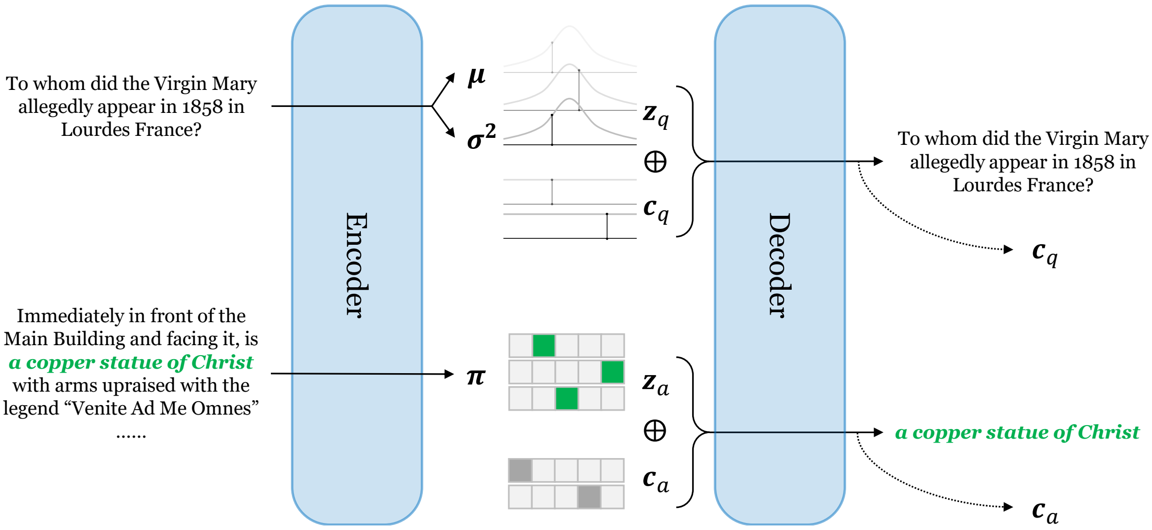

Figure 1: The overview of VOLTA.

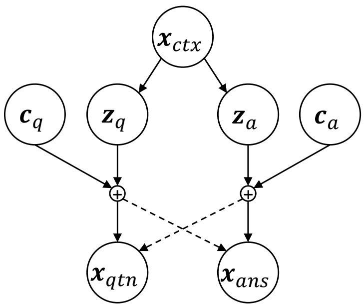

Figure 2: The graphical model for VOLTA.

Lower Bound (ELBO) objective:

$$\begin{aligned} \text{ELBO}(\boldsymbol{x}) &= \mathbb{E}_{q_\theta(z|\boldsymbol{x})}[\log p_\phi(\boldsymbol{x}|z)] \\ &\quad - D_{\text{KL}}(q_\theta(z|\boldsymbol{x}) \parallel p(z)) \\ &=: -\mathcal{L}_{\text{AE}}(\boldsymbol{x}) - \mathcal{L}_{\text{REG}}(\boldsymbol{x}) \end{aligned} \tag{2}$$

where we define the likelihood as the Autoencdoer (AE) reconstruction loss and the KL divergence as the regularization loss; the minus signs in front of the losses are because of the fact that we maximize the ELBO but minimize the losses.

The AE reconstruction loss will be introduced later in Section 3.4 because it involves the decoding step. The KL divergence can be used to regularize the posterior distributions $q_\theta(z|x)$ with the prior distribution $p(z)$. The KL divergence of a continuous latent variable is:

$$\begin{aligned} &D_{\text{KL}}(q_\theta(z|x) \parallel p(z)) \\ &= \log \frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}, \end{aligned} \tag{3}$$

where we assume that $p(z)$ is $\mathcal{N}(\mu_p, \sigma_p^2)$ and $q_\theta(z|x)$ is $\mathcal{N}(\mu_q, \sigma_q^2)$. The KL divergence of a discrete latent variable is:

$$D_{\text{KL}}(q_\theta(z|x) \parallel p(z)) = \sum_{i=1}^{k} q_i \log \frac{q_i}{p_i}, \tag{4}$$

where the event probabilities of the prior $p(z)$ are $(p_1, \ldots, p_k)$ and those of the posterior $q_\theta(z|x)$ are $(q_1, \ldots, q_k)$. The derivation of those results can be found in Appendix A.2, A.3

3.3 Latent Codes

In addition to latent variables, we add latent codes to inject controllability into the model, which was originally proposed in InfoGAN (Chen et al., 2016) from the field of Computer Vision. There are also two types of latent codes: continuous and discrete. Continuous latent codes can follow either the uniform distribution or the Gaussian distribution, while discrete latent codes can still use the categorical distribution. In our model, we draw $c_q \sim \text{Uni}(-1, 1)$ and $c_a \sim \text{Cat}(\boldsymbol{\rho})$, where $\text{Uni}(\cdot)$ is the uniform distribution; $\text{Cat}(\cdot)$ is the categorical distribution with parameters $\boldsymbol{\rho} = \frac{1}{k}\mathbf{1}$ that uses the same number of categories k as the discrete latent variables, because they will be concatenated together.

To prevent the model from ignoring the latent codes, we encourage the model to recover the input latent code at the generation step. To achieve that, we add the Variational Mutual Information Maximization (VMIM) objective (Chen et al., 2016):

$$
\begin{aligned}
& I(c; f_\phi(z, c)) \\
&= H(c) + \mathbb{E}_{x\sim f_\phi(z,c)}\Big[D_{\mathrm{KL}}\big(P(\cdot|x) \parallel P_\phi(\cdot|x)\big) \\
&\quad + \mathbb{E}_{c'\sim P(c|x)}\big[\log P_\phi(c'|x)\big]\Big] \qquad (5) \\
&\geq H(c) + \mathbb{E}_{x\sim f_\phi(z,c)}\Big[\mathbb{E}_{c'\sim P(c|x)}\big[\log P_\phi(c'|x)\big]\Big] \\
&=: H(c) + \mathcal{L}_{\mathrm{VMIM}}(c)
\end{aligned}
$$

Because the posterior $P(c|x)$ is difficult to obtain, an auxiliary distribution $P_\phi(c|x)$ based on f_ϕ is added to approximate $P(c|x)$. The entropy of latent codes $H(c)$ is a constant and thus it is excluded from the VMIM objective. The derivation of this objective is included in Appendix A.4.

In practice, a fully-connected layer is added to the decoder for each latent code whose objective is to recover the original latent code:

$$
\begin{aligned}
\mu_c, \sigma_c^2 &= \mathrm{MLP}(f_\phi(\boldsymbol{z}_q \oplus c_q, \boldsymbol{x}_{ctx})) \\
\boldsymbol{\rho}_c &= \mathrm{MLP}(f_\phi(\boldsymbol{z}_a \oplus c_a, \boldsymbol{x}_{ctx})) \\
\mathcal{L}_{\mathrm{VMIM}}(c_q) &= \log P(c_q; \mu_c, \sigma_c^2) \\
\mathcal{L}_{\mathrm{VMIM}}(c_a) &= \log P(c_a; \boldsymbol{\rho}_c).
\end{aligned} \qquad (6)
$$

We have two channels to pass the latent variable information to the decoder. One channel is to use a linear layer to obtain a latent embedding that is added to the word embedding, along with positional encoding; the other channel is to generate a latent embedding for each Transformer decoder block of the decoder, and those latent embeddings are treated as the past information for the decoder blocks. These two channels are termed "embedding" and "memory" in Optimus.

3.4 Question & Answer Generation

To reconstruct the original questions, the Autoencoder is trained as a language model in an auto-regressive manner, which predicts the next token given all previous tokens.

$$
\begin{aligned}
p_\phi(x_t) &= \mathrm{MLP}(f_\phi(\boldsymbol{z}_a \oplus \boldsymbol{c}_a, \boldsymbol{z}_q \oplus \boldsymbol{c}_q, \boldsymbol{x}_{<t})) \\
p_\phi(\boldsymbol{x}_{qtn}) &= \prod_{t=1}^{n} p(x_t|\boldsymbol{x}_{<t})
\end{aligned} \qquad (7)
$$

where $\boldsymbol{c}_a$ is a vector that contains multiple independent categorical latent codes, and $\boldsymbol{c}_q$ is a vector that contains multiple independent uniform latent codes; p_ϕ is conditioned on $\boldsymbol{x}_{ctx}$, which is omitted for brevity.

Therefore, the question reconstruction loss is a cross-entropy loss over the vocabulary with respect to all question tokens:

$$
\mathcal{L}_{\text{Qtn-AE}}(\boldsymbol{x}) = \sum_{t=1}^{n} \mathrm{CE}(p_\phi(x_t|\boldsymbol{x}_{<t}), y_t). \qquad (8)
$$

Because SQuAD answers are annotated by two indices, one for the start word and the other for the end word. When the model tries to reconstruct the answer, it also predicts those two indices. Hence, the answer reconstruction loss is:

$$
\begin{aligned}
p_{start}(\boldsymbol{x}_{ctx}) &= \mathrm{MLP}(f_\phi(\boldsymbol{z}_a \oplus \boldsymbol{c}_a, \boldsymbol{x}_{ctx})) \\
p_{end}(\boldsymbol{x}_{ctx}) &= \mathrm{MLP}(f_\phi(\boldsymbol{z}_a \oplus \boldsymbol{c}_a, \boldsymbol{x}_{ctx})). \\
\mathcal{L}_{\text{Ans-AE}}(\boldsymbol{x}) &= \mathrm{CE}(p_{start}(\boldsymbol{x}_{ctx}), y_{start}) \\
&\quad + \mathrm{CE}(p_{end}(\boldsymbol{x}_{ctx}), y_{end}),
\end{aligned} \qquad (9)
$$

where $\boldsymbol{c}_a$ is a vector that contains multiple independent categorical latent codes; $\oplus$ is the concatenation operation; y_{start} and y_{end} are the true answer span; $\mathrm{CE}(\cdot)$ is the cross-entropy loss.

Therefore, the overall Autoencdoer reconstruction loss is the sum of both AE losses:

$$
\mathcal{L}_{\text{AE}}(\boldsymbol{x}) = \mathcal{L}_{\text{Qtn-AE}}(\boldsymbol{x}) + \mathcal{L}_{\text{Ans-AE}}(\boldsymbol{x}) \qquad (10)
$$

3.5 QA Mutual Information

In addition, we also want to enforce the mutual information between the generated question and answer (QAMI). As in Info-HCVAE (Lee et al., 2020), we base this QAMI objective on Jensen-Shannon Divergence:

$$
\begin{aligned}
g(q, a) &= \sigma(f_\phi(q)^T \boldsymbol{W} f_\phi(a)) \\
\mathcal{L}_{\text{QAMI}}(\boldsymbol{x}) &= \mathbb{E}[\log g(q, a)] \\
&\quad + \frac{1}{2}\mathbb{E}[\log(1 - g(\tilde{q}, a))] \\
&\quad + \frac{1}{2}\mathbb{E}[\log(1 - g(q, \tilde{a}))] \\
&\leq I(q, a),
\end{aligned} \qquad (11)
$$

where q is the embedding of the question by f_ϕ and a is the embedding of the answer; $\tilde{q}$ is a negative question sample and $\tilde{a}$ is a negative answer sample. $g(\cdot)$ adds a bilinear layer on top of f_ϕ and classifies whether the input question and answer is a true pair of QA.

Therefore, by Eq. (2)(6)(10)(11), we have the overall loss being:

$$
\begin{aligned}
\mathcal{L}_{\text{ELBO}}(x) &= \mathcal{L}_{\text{AE}}(\boldsymbol{x}) + \beta \mathcal{L}_{\text{REG}}(\boldsymbol{x}) \\
&\quad + \mathcal{L}_{\text{VMIM}}(\boldsymbol{c}) + \mathcal{L}_{\text{QAMI}}(\boldsymbol{x})
\end{aligned} \qquad (12)
$$

	Similarity to Reference						Diversity				
	BLEU-1↑	BLEU-2↑	BLEU-3↑	BLEU-4↑	MTR ↑	RG-L ↑	Dist-1↑	Dist-2↑	Dist-3↑	Dist-4↑	S-BLEU ↓
GPT-2 (Radford et al., 2019)	51.456	35.610	26.608	20.461	23.109	48.983	**8.408**	38.472	61.608	73.627	33.042
Info-HCVAE (Lee et al., 2020)	48.167	30.200	20.522	14.321	19.865	43.918	6.997	33.473	57.242	71.681	32.658
VOLTA (**ours**)	33.243	16.025	9.346	5.814	11.944	31.257	7.894	38.697	**65.488**	**80.793**	**29.579**
Small $\boldsymbol{z}_q$	32.740	16.064	9.543	5.974	11.621	31.798	7.420	34.191	58.127	73.210	33.435
Small $\boldsymbol{z}_a$	33.339	16.056	9.405	5.889	21.620	46.272	7.601	38.168	65.065	80.480	29.849
Large $\boldsymbol{z}_q$	33.055	16.364	9.896	6.408	11.928	31.755	7.245	33.081	55.647	69.922	37.539
Large $\boldsymbol{z}_a$	35.006	17.817	10.899	7.123	12.465	33.198	7.004	31.237	51.695	64.220	43.233
W/o $\boldsymbol{c}_q$ & $\boldsymbol{c}_a$	33.677	17.048	10.426	6.806	12.366	31.790	7.870	37.073	61.864	76.316	33.094
QG only	50.159	32.853	23.424	17.244	21.620	46.272	7.983	**39.248**	65.080	78.438	29.591

Table 2: Performance comparison and ablation study. "MTR" means METEOR, "RG-L" means ROUGE-L, "Dist-k" means Distinct-k, and "S-BLEU" means Self-BLEU.

where $\boldsymbol{c}$ represents all the independent continuous and discrete latent codes; β is the coefficient for the KL divergence losses. Because of the KL vanishing issue (Bowman et al., 2016) where the decoder ignores the latent variables, we also use a linear annealing schedule for β (Li et al., 2020) and limit its maximal value to 0.1 (Lee et al., 2020).

4 Experiments

4.1 Implementation Details

We use the "GPT2-base" model as the backbone network. Our model uses the following configuration if not otherwise specified: the number of Gaussian latent variables is 32; the number of categorical latent variables is 20 and each of them has 10 categories; 4 uniform latent codes are added alongside with the Gaussian latent variables and together they are used to handle the information from questions; 5 categorical latent codes are concatenated to the categorical latent variables and they are dedicated to process answer embeddings. The model is trained with a learning rate of 5×10^{-5} for 20 epochs. The annealing schedule for β includes an increasing phase that spans 25% of the total training time, from 0 up to the maximal value of 0.1, which is maintained for the rest of the training duration. The experiments are conducted using 4 TITAN V GPUs.

4.2 Question Generation Diversity

We first test the question generation quality with BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004) on the SQuAD dataset (Rajpurkar et al., 2016, 2018). The BLEU score measures the similarity between generated sentences and the reference sentences based on n-grams. METEOR (Banerjee and Lavie, 2005) uses the harmonic mean of the precision and recall of unigrams instead, and it takes more factors into consideration, such as stemming and synonymy. ROUGE-L (Lin, 2004) primarily considers the longest common subsequences.

As we can see in Table 2, because the VAE framework perturbs the latent variables, the generated questions divert from the reference questions. This indicates that our model generation is less anchored at the ground truth questions and thus more diverse. GPT-2 is not designed to generated answer spans and thus it generates questions with ground truth answers.

To quantify the diversity of the generated questions, we use two diversity measures: Distinct-k (Li et al., 2016) and Self-BLEU (Zhu et al., 2018). Distinct-k is the number of distinct k-grams divided by the total number of generated words. Self-BLEU regards every generated sentence as hypothesis and the other sentences as reference to calculate the BLEU score with respect to the hypothesis sentence; then the average BLEU score over all generated sentences is the Self-BLEU of the document. If the generated sentences in the document are diverse, the Self-BLEU score will be low. As shown in Table 2, our model has higher overall diversity.

4.3 Ablation Study

We experiment with different configurations of our model, as shown in Table 2. "small $\boldsymbol{z}_q$": the number of Gaussian latent variables is reduced from the default 32 to 8 while all other components are unchanged; "small $\boldsymbol{z}_a$": 5 categorical latent variables are used instead of 20; "large $\boldsymbol{z}_q$": the model uses 64 Gaussian latent variables; "large $\boldsymbol{z}_a$": there are 40 categorical latent variables in the model; "w/o $\boldsymbol{c}_q$ & $\boldsymbol{c}_a$": no latent codes are added; "QG only": the model does not generate answers and the questions are generated based on ground truth answer spans.

The experimental results show that when the latent variables are too small, the encoded latent information in them might be insufficient for the decoder; but when the latent variables are too large, the perturbation of the Gaussian distribution or the categorical distribution may compound and distort the latent information too much. By removing the latent codes, we can see the diversity metrics drop. This indicates that the latent codes also improve the model diversity. When the model does not generate answers, the similarity-to-reference metrics are much better. Because the generated answers are very different from the original ones and the questions are generated with respect to the generated answers, adding answer generation can pull the generated questions away from the reference questions, which improves the diversity while sacrificing the similarity to the reference questions.

4.4 Downstream Task Analysis

Although with the two diversity metrics, Distinct-k (Li et al., 2016) and Self-BLEU (Zhu et al., 2018), we were able to show that our model generates more diverse questions. But a model can achieve good results for those two metrics if it merely generates completely random tokens. Therefore, we use two additional metrics, QAE and R-QAE, based on an auxiliary downstream task of question answering (QA) to show that the generated questions are diverse and non-arbitrary sequences.

	QAE↑		R-QAE↓	
	EM	F1	EM	F1
GPT-2 (Radford et al., 2019)	56.6382	68.6164	67.3124	79.4297
Optimus (Li et al., 2020)	**58.2745**	**70.5103**	67.0479	78.8968
Info-HCVAE (Lee et al., 2020)	56.9543	68.5626	40.2104	58.7262
VOLTA (**ours**)	56.9357	68.6692	**19.8872**	**31.0355**

Table 3: Quality-diversity trade-off of QA pair generation.

QAE Zhang and Bansal (2019) proposed **Q**uestion-**A**nswering-based **E**valuation (QAE) to measure the quality of the generated question-answer pairs. To measure the QAE of a model, one need to follow four main steps: (a) sample some unlabeled Wikipedia paragraphs with pre-extracted answer spans from HarvestingQA dataset; (b) make the QG model that we want to measure act as an "annotator" to generate a question for each answer span, which results in a synthetic QA dataset; (c) train a separate QA model using this synthetic QA dataset; (d) use the performance of the trained QA model on the original SQuAD development set (Rajpurkar et al., 2016, 2018) as the evaluation for this QG model, which includes two measurements, exact match (EM) and F1 (Rajpurkar et al., 2016, 2018). QAE primarily measures the quality of the generated questions. If the generated questions are composed of random tokens, the trained QA model will perform badly on the development set of SQuAD. The BERT model (Devlin et al., 2019) is used as the QA model.

R-QAE If we train a QA model using the original SQuAD training set but we test the trained QA model on a synthetic QA test set, the performance is expected to be low when the synthetic dataset is diverse. The reason is that when the generated test dataset has more diversity and out-of-distribution QA pairs, the QA model is expected to perform badly. Because the evaluated QG model is used to annotate the test set in R-QAE rather than the training set in QAE, it is named Reverse-QAE, or R-QAE for short (Lee et al., 2020).

As we can observe in Table 3, our model does not sacrifice the question generation quality while achieving better diversity than the baselines.

4.5 Diverse & Controllable Generation

Our model architecture enables two main ways to control the generation process. One is from the VAE framework (Kingma and Welling, 2014), which provides the latent variables that can be used to interpolate between source and target examples. The other one is based on adjusting the latent codes from InfoGAN (Chen et al., 2016). Unlike the latent variables, latent codes are independent of the input context.

Latent Variable Diversity Given a context, we can generate different z_q and z_a because of the nature of VAE. Therefore, we can generate different QA pairs from the same context. The shortcoming of this approach is that the user has no control over the latent variables. The latent variables are completely dictated by the encoder and the randomness of the learned latent distributions. An example of the QA pairs generated for a given context is illustrated in Table 1.

Latent Variable Interpolation By encoding two contexts (can be the same context) into two sets of latent variables, we can obtain new latent variables by linearly interpolating between them. However, this method suffers from two drawbacks: first, when we get two sets of latent variables from two different contexts, they might be very dissimilar to each other and the semantics of the interpolated points is not clear; second, it is also not reasonable to interpolate between the two categorical latent variables. An example of interpolated results can be found in Table 4.

Context The university is the major seat of the Congregation of Holy Cross (albeit not its official headquarters, which are in Rome). Its main seminary, Moreau Seminary, is located on the campus across St. Joseph lake from the Main Building. ......
Q1 What catholic denomination is the university of new haven located in?
Q2 What is the main campus of moreau seminary?
Q3 What religious institution is located on the campus of moreau seminary?
Q4 What former retreat center is located near the grotto?
Q5 What religious denomination does the moreau seminary belong to?
Q6 What is the oldest building on campus?
Q7 What is the main seminary in the university of kansas?
Q8 What is the main seminary of the college?
Q9 What retreat center is located near the grotto?

Table 4: An example of interpolating between latent variables for question generation.

Latent Code Controllability Unlike latent variables that are highly dependent on the inputs, latent codes can be set freely regardless of what the context is. Because they are passed to the decoder alongside with the latent variables, they do not degrade the information contained in the latent variables. They add more dimensions for controlling the output, besides the controllability from the latent variables. As we can see in Table 5 and Table 6, the continuous latent codes can adjust question generation while the discrete latent codes can be used to change the generated answers.

Context Holy Cross Father John Francis O'Hara was elected vice-president in 1933 and president of Notre Dame in 1934. During his tenure at Notre Dame, he brought numerous refugee intellectuals to campus; ......
Q1 ($c_q = -0.8$) What was O'Hara's first name?
Q2 ($c_q = -0.6$) Who was elected vice president in 1933?
Q3 ($c_q = -0.0$) What was O'Hara's title prior to becoming vice president?
Q4 ($c_q = +0.4$) What was O'Hara's first title?
Answer John Francis O'Hara

Table 5: Continuous latent code for controlling question generation.

Context ......During his 13 years the Irish won three national championships, had five undefeated seasons, won the Rose Bowl in 1925, and produced players such as George Gipp and the "Four Horsemen". ......
A1 ($c_a = 0$) five
A2 ($c_a = 3$) 1925
A3 ($c_a = 7$) three

Table 6: Discrete latent code for controlling answer generation.

4.6 Latent Variable Visualization

To visualize how latent variables are distributed in the latent space, we use t-SNE to plot latent variables of questions in a 2D space. It is compared with the GPT-2 embeddings for the same set of questions. As we can observe in Figure 3, GPT-2 returns the same embeddings for a given question while our model is able to encode a question into multiple different latent variables that follows the Gaussian distribution. Those distinct latent variables for a question then can be used to generated various questions after being handed to the decoder, which increases the diversity of our model.

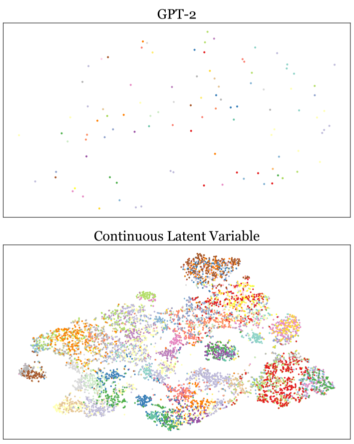

Figure 3: T-SNE visualization of question embeddings by GPT-2 and the latent variables by our model.

5 Conclusion

We developed a model named VOLTA that merges the power of Transformer models with the diversity from the VAE framework. The latent variables diversify the generated questions and answers. In addition, we all latent codes from InfoGAN to inject more dimensions of controllability. Both quantitative and qualitative experiments were carried out to show that our model indeed improves in diversity and controllability.

References

Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2018. Variational attention for sequence-to-sequence models. In *COLING*, pages 1672–1682. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*, pages 10–21. ACL.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *ICLR (Poster)*. OpenReview.net.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*, pages 489–500. Association for Computational Linguistics.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM*, 63(11):139–144.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *NeurIPS*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR (Poster)*. OpenReview.net.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

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.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324.

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional vaes. In *ACL*, pages 208–224. Association for Computational Linguistics.

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 *ACL*, pages 7871–7880. Association for Computational Linguistics.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *EMNLP (1)*, pages 4678–4699. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*, pages 110–119. The Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020. You impress me: Dialogue generation via mutual persona perception. In *ACL*, pages 1417–1427. Association for Computational Linguistics.

Ali Lotfi-Rezaabad and Sriram Vishwanath. 2020. Learning representations by maximizing mutual information in variational autoencoders. In *ISIT*, pages 2729–2734. IEEE.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR (Poster)*. OpenReview.net.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. SOLOIST: building task bots at scale with transfer learning and machine teaching. *Trans. Assoc. Comput. Linguistics*, 9:907–824.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL (2)*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org.

Zach Wolpe and Alta de Waal. 2019. Autoencoding variational bayes for latent dirichlet allocation. In *FAIR*, volume 2540 of *CEUR Workshop Proceedings*, pages 25–36. CEUR-WS.org.

Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, and Bill Dolan. 2021. A controllable model of grounded response generation. In *AAAI*, pages 14085–14093. AAAI Press.

Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020a. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *IJCAI*, pages 3997–4003. ijcai.org.

Xuerong Xiao, Swetava Ganguli, and Vipul Pandey. 2020b. Vae-info-cgan: generating synthetic images by combining pixel-level and feature-level geospatial conditional inputs. In *IWCTS@SIGSPATIAL*, pages 1:1–1:10. ACM.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: controllable story generation with external knowledge using large-scale language models. In *EMNLP (1)*, pages 2831–2845. Association for Computational Linguistics.

Fei Ye and Adrian G. Bors. 2021. Infovaegan: Learning joint interpretable representations by information maximization and maximum likelihood. In *ICIP*, pages 749–753. IEEE.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *EMNLP/IJCNLP (1)*, pages 2495–2509. Association for Computational Linguistics.

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

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 *SIGIR*, pages 1097–1100. ACM.

A Appendix

A.1 Basic Definitions

Information is defined as:

$$I(X) = -\log P(X) = \log \frac{1}{P(X)}.$$

Entropy is defined as:

$$\begin{aligned} H(X) =& \mathbb{E}[I(X)] \\ =& \mathbb{E}[-\log(P(X))] \\ =& -\int p(x)\log p(x)\mathrm{d}x \\ H(X|Y) =& \mathbb{E}_{X,Y}[-\log \mathrm{P}(X|Y)] \\ =& -\int f(x,y)\log f(x|y)\mathrm{d}x\mathrm{d}y, \end{aligned}$$

where $p(x,y)$ is the probability mass function of a discrete distribution, whereas $f(x,y)$ is the probability density function of a continuous distribution.

Then mutual information is:

$$\begin{aligned} & I(X;Y) \\ =& D_{\mathrm{KL}}(P(X,Y) \parallel P(X)P(Y)) \\ =& \int p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\mathrm{d}x\mathrm{d}y \\ =& -\int p(x,y)\log p(y)\mathrm{d}x\mathrm{d}y \\ &+ \int p(x,y)\log\frac{p(x,y)}{p(x)}\mathrm{d}x\mathrm{d}y \\ =& -\int p(y)\log p(y)\mathrm{d}y \\ &+ \int p(x,y)\log p(y|x)\mathrm{d}x\mathrm{d}y \\ =& H(Y) - H(Y|X) \\ =& H(X) - H(X|Y), \end{aligned}$$

because Kullback–Leibler divergence is defined to be:

$$\begin{aligned} D_{\mathrm{KL}}(Q \parallel P) =& H(Q,P) - H(Q) \\ =& \mathbb{E}_Q[-\log \mathrm{P}(X)] - \mathbb{E}_Q[-\log \mathrm{Q}(X)] \\ =& \int q(x)\log\frac{q(x)}{p(x)}\mathrm{d}x \\ \geq& 0, \end{aligned}$$

where $H(Q,P)$ is the cross entropy of Q and P.

A.2 Optimus (β-VAE)

In Optimus (Li et al., 2020; Kingma and Welling, 2014), we assume a normal distribution for a continuous latent variable:

$$\begin{aligned} f(x) =& \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \\ \log f(x) =& -\log\sigma\sqrt{2\pi} - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \\ =& -\log\sigma - \frac{1}{2}\log 2\pi - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \\ =& -\frac{1}{2}\log\sigma^2 - \frac{1}{2}\log 2\pi - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2. \end{aligned}$$

We want $q(z|x) = N(\mu_q, \sigma_q^2)$ and the prior, $p(z) = N(\mu_p, \sigma_p^2) = N(0,1)$, to be close

$$\begin{aligned} & D_{\mathrm{KL}}(Q \parallel P) \\ =& -\int q(z)\log p(z)dz + \int q(z)\log q(z)dz \\ =& \left(\frac{1}{2}(\log 2\pi\sigma_p^2) + \frac{\sigma_q^2 + (\mu_q-\mu_p)^2}{2\sigma_p^2}\right) \\ & - \frac{1}{2}(1+\log 2\pi\sigma_q^2) \\ =& \frac{1}{2}(\log\frac{\sigma_p^2}{\sigma_q^2}) + \frac{\sigma_q^2 + (\mu_q-\mu_p)^2}{2\sigma_p^2} - \frac{1}{2} \\ =& \frac{1}{2}\log\left(\frac{\sigma_p}{\sigma_q}\right)^2 + \frac{\sigma_q^2 + (\mu_q-\mu_p)^2}{2\sigma_p^2} - \frac{1}{2} \end{aligned}$$

The mutual information between z and $z|x$ is

$$I(z,x) = H(z) - H(z|x),$$

where the negative entropy for normal distribution is (n_z is the dimension of latent variable z):

$$\begin{aligned} -H(z|x) =& \mathbb{E}_{Q(z|x)}[\log(Q(z|x))] \\ =& -\int q(z)\log q(z)\mathrm{d}z \\ =& -\frac{1}{2}(1+\log 2\pi\sigma_q^2) \\ =& -\frac{1}{2}(1+\log 2\pi + \log\sigma_q^2) \\ =& -\frac{1}{2}\log 2\pi - \frac{1}{2}(1+\log\sigma_q^2) \end{aligned}$$

$$
\begin{aligned}
H(z) &= \mathbb{E}_{q(z)}[-\log q(z)] \\
&= -\int q(z)\left(\log \sigma_q\sqrt{2\pi} + \frac{1}{2}\left(\frac{z-\mu_q}{\sigma_q}\right)^2\right)\mathrm{d}x \\
&= -\int q(z)\log \sigma_q\sqrt{2\pi}\mathrm{d}x \\
&\quad -\int q(z)\frac{1}{2}\left(\frac{z-\mu_q}{\sigma_q}\right)^2\mathrm{d}x \\
&= -\mathbb{E}_{q(z)}[\log \sigma_q\sqrt{2\pi}] - \mathbb{E}_{q(z)}\left[\frac{1}{2}\left(\frac{z-\mu_q}{\sigma_q}\right)^2\right] \\
&= -\log \sigma_q\sqrt{2\pi} - \mathbb{E}_{q(z)}\left[\frac{1}{2}\left(\frac{z-\mu_q}{\sigma_q}\right)^2\right] \\
&= -\log \sigma_q\sqrt{2\pi} - \frac{1}{2}\left(\frac{\mathbb{E}_{q(z)}\left[(z-\mu_q)^2\right]}{\sigma_q^2}\right) \\
&= -\frac{1}{2}\log \sigma_q^2 - \frac{1}{2}\log 2\pi - \frac{1}{2}\frac{(z-\mu_q)^2}{\sigma_q^2},
\end{aligned}
$$

where $\mathbb{E}_{q(z)}\left[(z-\mu_q)^2\right]$ is simply the deviation of a single sample z from the mean μ_q.

A.3 Info-HCVAE

According to Info-HCVAE (Lee et al., 2020), some inputs are better suited to be encoded into discrete latent variables. In this case, we can make use of the categorical distribution:

$$f(x = i \mid \boldsymbol{p}) = p_i,$$

where the event probabilities $\boldsymbol{p} = (p_1, \ldots, p_k)$ and $\sum_{i=1}^{k} p_i = 1$; $k > 0$ is the number of categories.

The Gumbel-Softmax distribution enables back-propagation through discrete distributions. The Gumbel distribution is:

$$\text{Gumbel}(\mu, \beta) = f(x; \mu, \beta) = \frac{1}{\beta}e^{-(z+e^{-z})},$$

where $z = \frac{x-\mu}{\beta}$.

To sample a category from the categorical distribution using the Gumbel-Max re-parametrization trick, one can follow:

$$\arg\max_i (G_i + \log p_i),$$

where $G_i \sim \text{Gumbel}(0, 1)$. $\arg\max$ can be made differentiable by approximating it with the softmax function:

$$y_i = \frac{\exp((G_i + \log p_i)/\tau)}{\sum_j \exp((G_j + \log p_j)/\tau)},$$

Given two categorical distributions P and Q, parameterized by $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively, the KL divergence between them is:

$$D_{\text{KL}}(Q \parallel P) = \sum_{i=1}^{k} q_i \log \frac{q_i}{p_i}.$$

A.4 InfoGAN

The input noise z is passed into the generator along with the latent code c: $G(z, c)$, where z is concatenated with c. Because the generator can simply ignore the latent code c, InfoGAN (Chen et al., 2016) adds Variational Mutual Information Maximization (VMIM) to maintain the mutual information between generated sample $x \sim G(z, c)$ and latent code c:

$$
\begin{aligned}
&I(c; G(z, c)) \\
=&H(c) - H(c|G(z, c)) \\
=&H(c) + \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] \\
=&H(c) + \mathbb{E}_{x\sim G(z,c)}\Big[\sum_{c'} p(c'|x)\log p(c'|x)\Big] \\
=&H(c) + \mathbb{E}_{x\sim G(z,c)}\Big[\sum_{c'} p(c'|x)(\log \frac{p(c'|x)}{q(c'|x)} \\
&\qquad + \log q(c'|x))\Big] \\
=&H(c) + \mathbb{E}_{x\sim G(z,c)}\Big[\sum_{c'} p(c'|x)\log \frac{p(c'|x)}{q(c'|x)} \\
&\qquad + \sum_{c'} p(c'|x)\log q(c'|x)\Big] \\
=&H(c) + \mathbb{E}_{x\sim G(z,c)}\big[D_{\text{KL}}(P(\cdot|x) \parallel Q(\cdot|x)) \\
&\qquad + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]\big] \\
\geq&H(c) + \mathbb{E}_{x\sim G(z,c)}\left[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]\right],
\end{aligned}
$$

Because the posterior $P(c|x)$ is hard to obtain, an auxiliary distribution $Q(c|x)$ is added to approximate $P(c|x)$, where Q is a neural network. In practice, the entropy of latent codes $H(c)$ is treated as a constant and omitted in the InfoGAN objective.

A.5 InfoVAE and InfoMax-VAE

The evidence lower bound (ELBO) of regular VAE is

$$
\begin{aligned}
&\mathcal{L}_{\text{ELBO}}(x) \\
=&\mathcal{L}_{\text{AE}}(x) + \mathcal{L}_{\text{REG}}(x) \\
=&\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x) \parallel p(z)) \\
\leq&\log p_\theta(x).
\end{aligned}
$$

InfoVAE (Zhao et al., 2019) and InfoMax-VAE (Lotfi-Rezaabad and Vishwanath, 2020) add mutual information to the loss:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(x) =& \mathcal{L}_{\text{AE}}(x) + \beta\mathcal{L}_{\text{REG}}(x) + \alpha I_q(x;z) \\ =& \mathbb{E}_{p_D(x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]] \\ & - \beta\mathbb{E}_{p_D(x)} D_{\text{KL}}(q_\phi(z|x) \parallel p(z)) \\ & - \alpha D(q_\phi(x;z) \parallel q(x)q_\phi(z)),\end{aligned}$$

Because $D(q_\phi(x;z) \parallel q(x)q_\phi(z))$ is usually intractable; thus, it can be approximated with any one of the following:

- KL divergence
- f-divergence (InfoMax)
- Donsker-Varadhan dual representation (InfoMax)
- Jensen Shannon divergence (AAE)
- Stein Variational Gradient
- Maximum-Mean Discrepancy