Applied Intelligence (2021) 51:9030–9037
https://doi.org/10.1007/s10489-021-02374-7

ORIGINAL SUBMISSION

SeqVAE: Sequence variational autoencoder with policy gradient

Ting Gao[1] · **Yidong Cui**[1] · **Fanyu Ding**[1]

Accepted: 23 March 2021 / Published online: 21 April 2021

Abstract

In the paper, we propose a variant of Variational Autoencoder (VAE) for sequence generation task, called SeqVAE, which is a combination of recurrent VAE and policy gradient in reinforcement learning. The goal of SeqVAE is to reduce the deviation of the optimization goal of VAE, which we achieved by adding the policy-gradient loss to SeqVAE. In the paper, we give two ways to calculate the policy-gradient loss, one is from SeqGAN and the other is proposed by us. In the experiments on them, our proposed method is better than all baselines, and experiments show that SeqVAE can alleviate the "post-collapse" problem. Essentially, SeqVAE can be regarded as a combination of VAE and Generative Adversarial Net (GAN) and has better learning ability than the plain VAE because of the increased adversarial process. Finally, an application of our SeqVAE to music melody generation is available online[1][2].

Keywords Sequence generation task · Variational autoencoder · Generative Adversarial net · SeqVAE

1 Introduction

The generative models plays an important role in unsupervised learning. It is widely adopted in tasks such as generating new samples or estimating the probability of sample occurrences. In recent years, the generative models under the influence of deep learning has developed rapidly. The generative models combined with the deep neural network are called "deep generative models", and they take advantage of the superior expression and high-dimensional modeling capabilities of the deep neural network. So far, the typical deep generative models include: CycleGAN [30] which has made progress in style transfer; WaveNet [21] which has made a breakthrough in the field of speech synthesis; and StyleGAN [15] which has achieved success in the field of image generation. Among the many deep generative models, the special attention required from GAN [10] and VAE [17] because there are some deep generation models that are their variants.

GAN is an implicit generative model whose network structure consists of a generator and a discriminator. In the process of training GAN, the generator aims to mix the spurious with the genuine, and the discriminator is used to identify authenticity. They are opposed to each other. Finally, we use the generator that has learned the data distribution to generate data. In many generation tasks, variants of GAN have achieved state-of-the-art performance. However, there are still some problems with GAN. GAN training is difficult and the training process is unstable. To address this, some studies have perfected the loss function of GAN (for example, WGAN [1]), and some studies are committed to modifying the network structure of GAN (for example, [29] designs a new generator). In addition, it is difficult to use GAN to generate sequences of discrete tokens, such as the text [14]. For this, SeqGAN [28] provides a solution based on the ideas in reinforcement learning.

The other model is VAE, its network structure is composed of encoder and decoder. In VAE, we are able to explicitly model the latent vector z and that is a clear advantage over GAN. There [6, 8, 11] are some interesting innovative applications about VAE. For example, interpolating between latent vectors and decoding points on the trajectory can generate realistic intermediate samples

[1] https://github.com/INotWant/MelodyGeneration_SeqVAE1

[2] https://github.com/INotWant/MelodyGeneration_SeqVAE2

✉ Yidong Cui
cyd@bupt.edu.cn

Ting Gao
tinggao@bupt.edu.cn

Fanyu Ding
nuoyi618@bupt.edu.cn

[1] College of Computing, Beijing University of Posts and Telecommunications, Beijing, China

[23]. However, VAE has a key problem, and the data it generates is often not as real as the one generated by GAN. In fact, VAE is optimized for the evidence lower bound (ELBO), which deviates from the true data distribution. In the paper, we try to modify objective function of VAE to reduce this deviation. A very natural idea is to add an adversarial process to the training of the VAE, and we do just that. Inspired by SeqGAN, we introduced policy gradient [25] for VAE, and finally got a combination model of VAE and GAN. So from the perspective of GAN, our work provides a solution for generating sequences of discrete tokens using GAN.

In the paper, we focus on the combination of recurrent VAE [5] and policy gradient. Recurrent VAE is a typical sequence generation model, which can be used for text generation, symbolic music generation, etc. Unlike the extensively researched tasks of generating continuous-valued data with a fixed dimensionality (such as images), modeling sequential data is less common [23]. This is mainly because an autoregressive decoder is required, and autoregressive decoder is prone to "posterior-collapse" problem when training. MusicVAE [5] use a hierarchical decoder to avoide the "pos-terior collapse" problem. In the paper, we find that adding the policy-gradient loss can effectively avoid this problem.

The paper is structured as the following: In Section 2, we present related work. In Section 3, we introduce the network structure and the optimization goals of SeqVAE. In Section 4, we show and analyze our experiments. In Section 5, we summarize the research and discuss future work.

2 Related work

Sequence generation task is a common generation task. Text generation, symbolic music generation, and machine translation are all sequence generation tasks. In recent years, sequence generation models have developed to some extent. Recurrent VAE is a common sequence generation model and it often uses recurrent neural networks (RNNs) [13] as a codec. RNNs facilitates the modeling of sequence data, but the training of RNNs has a problem. The most common way to train RNNs is to maximize the token sequence that appears in the training set. [4] pointed out that this training method is biased due to the difference between the training environment and the generation environment, and proposed a training strategy called scheduled sampling (SS). After that, [14] pointed out the problems of the objective function under SS. In addition, GAN that has been successful in many generation fields is also considered for sequence generation task. However, [9] pointed out that it is difficult to generate sequences of discrete tokens using GAN. Therefore, some scholars study how to use GAN to generate sequences of discrete tokens, and some scholars continue to research recurrent VAE.

[2] proposed to describe the sequence generation process as a sequential decision process, which points out a direction for GAN-based sequence generation research. Based on this, SeqGAN [28] that introduces policy gradient is the first work extending GAN to generate sequences of discrete tokens.

At the same time, research on recurrent VAE has also made progress. [24] proposed a recurrent VAE for generating text, in which they replaced RNNs with feedforward neural networks and convolutional neural networks. [23] proposed a hierarchical recurrent VAE, which can effectively avoid the "posterior-collapse" problem. Our work is based on the research of recurrent VAE and we still use RNNs as codecs. The difference is that, inspired by the work of SeqGAN, we introduced policy gradient for recurrent VAE to enhance the learning ability.

From the perspective of the network structure, SeqVAE is a combination of VAE and GAN. Previous studies have also tried to combine them. [27] based on the work of SeqGAN, VGAN is proposed for text generation. The way they combine policy gradient is completely in accordance with SeqGAN. Earlier attempts, such as CVAE-GAN [3], were not made under the idea of reinforcement learning, which is different from our starting point.

3 Sequence variational autoencoder

In SeqGAN, the policy gradient is introduced to solve the problem that GAN is difficult to generate sequences of discrete tokens, and the GAN's ability to learn fragments in sequences is also enhanced [28]. The recurrent VAE introduces policy gradient is to optimize the objective function to reduce the deviation of ELBO from the real data distribution.

In this section, we first briefly introduce the recurrent VAE. And then we show the modification to recurrent VAE by introducing the network structure of SeqVAE in detail. We elaborate two different ways to calculate the policy-gradient loss at the end.

3.1 recurrent VAE

In the recurrent VAE, firstly, the encoder $q_\lambda(z|x)$ uses RNNs to process the input sequence $x = \{x_1, x_2, \ldots, x_T\}$, and it outputs the distribution of latent vector z according to the state at the last time step. Secondly, we sample a z from the distribution obtained in the previous step. Finally, the decoder $p_\theta(x|z)$, whose initial state is initialized according to the latent vector z, generates a reconstructed sequence

$y = \{y_1, y_2, \ldots, y_T\}$ [5]. Based on variational inference, recurrent VAE trains by maximizing the ELBO:

$$ELBO = \mathbb{E}_{z\sim q_\lambda(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\lambda(z|x)||p(z)) \leq \log p(x) \tag{1}$$

where $KL(\cdot||\cdot)$ is the KL-divergence. In practice, the prior distribution $p(z)$ is unknown and we usually assume that it follows a standard normal distribution.

When analyzing (1), we can divide the ELBO into two parts: $\mathbb{E}_{z\sim q_\lambda(z|x)}[-\log p_\theta(x|z)]$ and $\mathrm{KL}(q_\lambda(z|x)||p(z))$. The previous item describes the distance between the reconstructed sequence and the original sequence and is usually called the reconstruction loss. The latter term describes the distance between the approximate posterior distribution and the prior distribution and is usually called the KL-divergence loss. During training, the two are opposite, so they need to be balanced. This situation still exists in SeqVAE, in order to alleviate this situation we learn from the work of [5] and [17]. [5] added hyperparameter weight α to the KL divergence in the ELBO. The modified ELBO is:

$$\mathbb{E}_{z\sim q_\lambda(z|x)}[\log p_\theta(x|z)] - \alpha\mathrm{KL}(q_\lambda(z|x)||p(z)) \tag{2}$$

[17] set a threshold τ for the KL-divergence loss. Only when the KL-divergence loss exceeds the set threshold τ will it play a role in back propagation. The modified ELBO is:

$$\mathbb{E}_{z\sim q_\lambda(z|x)}[\log p_\theta(x|z)] - \max(\mathrm{KL}(q_\lambda(z|x)||p(z)) - \tau, 0) \tag{3}$$

These two strategies have achieved good results, so they still be used in SeqVAE.

3.2 Network structure of SeqVAE

Figure 1 shows the network structure of SeqVAE, from which it can be seen that it contains the structure of SeqGAN. Therefore, we think it is a combination of VAE and GAN. This combination makes SeqVAE's training add the adversarial process, so SeqVAE has a higher ability to learn data.

As can be seen from Fig. 1, SeqVAE adds a new decoder and discriminator. The decoder $p_\beta(x|z)$ is exactly the same as the decoder $p_\theta(x|z)$, except that decoder $p_\beta(x|z)$ does not use back propagation to update the parameters, but uses the update rate r to update as follows:

$$\beta \leftarrow (1-r)\beta + r\theta \tag{4}$$

The decoder $p_\beta(x|z)$ is equivalent to the generator $G_\beta(x)$ in SeqGAN and is used for searching. The discriminator $d_\phi(x)$ [10] provides guidance for improving the decoder $p_\theta(x|z)$. The discriminator $d_\phi(x)$ input sequence $i = (i_1, i_2, \ldots, i_T)$ outputs the probability that the sequence i comes from the real world, and this probability is used as a reward signal of reinforcement learning. We train the discriminator $d_\phi(x)$ with real samples x' and samples f' generated by the decoder $p_\theta(x|z)$. The added decoder $p_\beta(x|z)$ and discriminator $d_\phi(x)$ finally serve the calculation of the policy-gradient loss.

3.3 SeqVAE via Policy Gradient 1

The reflection of policy gradient in SeqVAE is the policy-gradient loss, which corresponds to pg_loss in Fig. 1. We provide two ways to calculate the policy-gradient loss. Let's start with the approach from SeqGAN.

As in SeqGAN, the objective of policy gradient is:

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1\in\mathcal{Y}} p_\theta(y_1|s_0) \cdot Q_{d_\phi}^{p_\theta}(s_0, y_1) \tag{5}$$

where R_T is the reward representing the entire sequence, $\mathcal{Y}$ is a candidate dictionary, $s_i = \{z, y_0, ..., y_i\}$ (y_0 is the start preplacer) and $Q_{d_\phi}^{p_\theta}(s, a)$ is the action-value function of a squence, i.e. the expected accumulative reward starting

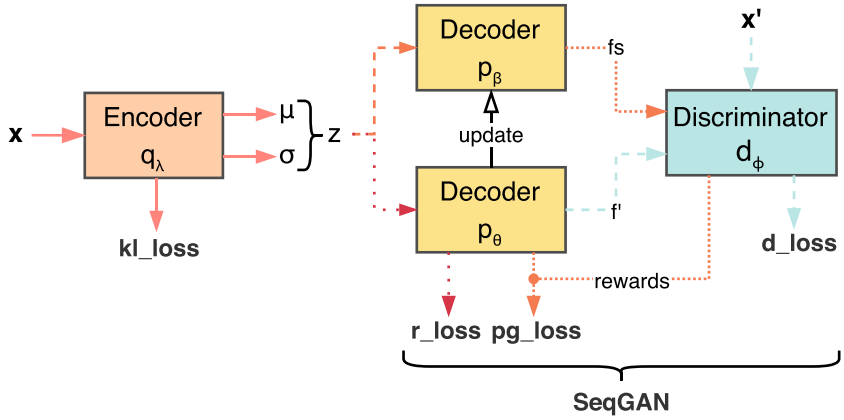

Fig. 1 Network structure of Sequence Variational Autoencoder model, SeqVAE. Samples x and x' are sampled from the training set. z is a latent vector. Samples fs and f' are generated by the decoder p_β and the decoder p_θ ,respectively. $rewards$ is obtained from the discriminator d_ϕ input fs. kl_loss is the KL-divergence loss, r_loss is the reconstruction loss, pg_loss is the policy-gradient loss, and d_loss is the loss of the discriminator $d_\phi(x)$

from state s, taking action a, and then following policy p_θ. REINFORCE algorithm [25] is still used, and Monte Carlo search is also needed in SeqVAE, we use the decoder $p_\beta(x|z)$ to complete it. Thus, we have:

$$\{Y_{1:T}^1, \ldots, Y_{1:T}^N\} = MC^{p_\beta}(Y_{1:t}; N) \tag{6}$$

$$Q_{d_\phi}^{p_\theta}(s_{t-1}, y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} d_\phi\left(Y_{1:T}^n\right), Y_{1:T}^n \in \mathrm{MC}^{p_\beta}(Y_{1:t}; N) & \text{for } t < T \\ d_\phi(Y_{1:t}) & \text{for } t = T \end{cases} \tag{7}$$

Equation (6) describes N Monte Carlo searches where $Y_{1:t}^n = (y_1, \ldots, y_t)$, $Y_{t+1:T}^n$ are sampled by the decoder $p_\beta(x|z)$ according to the current state. In this case, a Monte Carlo search is equivalent to the application of the Teacher Force to some extent. (7) gives the solution of action-value functions. After a series of derivations, the gradient of the objective function is obtained:

$$\nabla_\theta J(\theta) \approx \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_t \sim p_\theta(y_t|s_{t-1})}\left[\nabla_\theta \log p_\theta(y_t|s_{t-1}) \cdot Q_{d_\phi}^{p_\theta}(s_{t-1}, y_t)\right] \tag{8}$$

Please refer to [28] for the detailed derivation process. This calculation method (described by (8)) borrows from SeqGAN.

3.4 SeqVAE via Policy Gradient 2

In this subsection, we discuss our proposed calculation method. We start with the objective function (5) of policy gradient. We calculate it in another way here:

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_y p_\theta(y|s_0) R(y) \tag{9}$$

where $p_\theta(y|s_0)$ is the probability that the decoder generates the sequence y when the start state is s_0 and $R(y)$ is the reward that sequence y generated by the decoder can get. According to the previous description, $R(y)$ can be obtained from $d_\phi(y)$. Then we can get:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_y p_\theta(y|s_0) d_\phi(y) \\ &= \sum_y \frac{p_\theta(y|s_0)}{p_\theta(y|s_0)} \nabla_\theta p_\theta(y|s_0) d_\phi(y) \\ &= \mathbb{E}\left[\nabla_\theta \log p_\theta(y|s_0) d_\phi(y)\right] \end{aligned} \tag{10}$$

It can be seen that in order to calculate the gradient of the objective function, we need to calculate $\nabla_\theta \log p_\theta(y|s_0)$. We focus on this, and the derivation can be obtained:

$$\begin{aligned} &\nabla_\theta \log p_\theta(y|s_0) \\ &= \nabla_\theta \log\left[\prod_{t=1}^{T} P(s_t|s_{t-1}, y_t) \cdot p_\theta(y_t|s_{t-1})\right] \\ &= \nabla_\theta\left[\sum_{t=1}^{T} \log P(s_t|s_{t-1}, y_t) + \sum_{t=1}^{T} \log p_\theta(y_t|s_{t-1})\right] \\ &= \sum_{t=1}^{T} \nabla_\theta \log p_\theta(y_t|s_{t-1}) \end{aligned} \tag{11}$$

Thus, the gradient of the policy-gradient loss is solved:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[d_\phi(y) \cdot \sum_{t=1}^{T} \nabla_\theta \log p_\theta(y_t|s_{t-1})\right] \tag{12}$$

It can be seen that when calculating the gradient of the objective function, the reward is for the entire sequence. This is different from (8), the reward is for each time step. Therefore, the Monte Carlo search used by the decoder $p_\beta(x|z)$ is not required.

In order to enhance the learning of the fragments in the sequence, we use the decoder $p_\theta(x|z)$ to decode the same latent vector z for many times during the training process. Their differences are reflected in the application of the Teacher Force, so multiple different decoding processes are equivalent to the multiple Monte Carlo searches described by (6). Thus, we have to reconsider the calculation of $\nabla_\theta \log p_\theta(y|s_0)$. We use the following formula to approximate:

$$\nabla_\theta \log p_\theta(y|s_0) \approx \sum_{t=1}^{len} \nabla_\theta \log p_\theta(y_t|s_{t-1}) \tag{13}$$

where len is the length of the Teacher Force used by decoding. Finally, we can approximate (12) by sampling methods:

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m} d_\phi\left(y^{(i)}\right) \cdot \sum_{t=1}^{i \cdot step} \nabla_\theta \log p_\theta\left(y_t^{(i)}|s_{t-1}\right) \tag{14}$$

where m is the number of times of decoding, $y^{(i)}$ is the sequence of the i-th decoding output, and $step$ describes the length of each increase using the Teacher Force.

In practice, it is found that just as Actor and Critic are interdependent in the Actor-Critic algorithm [18], decoder $p_\theta(x|z)$ and discriminator $d_\phi(x)$ also are strong interdependence, which is not conducive to convergence.

In order to solve this problem, we learn from the work of modifying DQN [19] in [20]. We add a copy of the decoder $p_\theta(x|z)$, and the decoder $p_\beta(x|z)$ just meets our requirements. Therefore, the network structure of SeqVAE remains unchanged. Note that although the decoder $p_\beta(x|z)$ still does the same thing, its role has changed. We use it instead of the decoder $p_\theta(x|z)$ to perform the multiple decoding process.

Algorithm 1 Sequence variational autoencoder.

Require: encoder q_λ; decoder p_θ; decoder-copy p_β; discriminator d_ϕ; a sequence dataset $\mathcal{S} = \{x^{(1)}, \ldots, x^{(i)}, \ldots\}$

Initialize q_λ, p_θ, d_ϕ with random weights λ, θ, ϕ
Pre-train q_λ, p_θ by (2) or (3)
$\beta \leftarrow \theta$
Generate negative samples using p_θ for training d_ϕ
Pre-train d_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Enter $x \sim \mathcal{S}$ to q_λ to get the latent vector z and kl_loss
    Update encoder q_λ via kl_loss
    Use p_β to produce $\{y^{(1)}, ..., y^{(m)}\}$ by (6) and then use d_ϕ to get corresponding rewards $\{d_\phi(y^{(1)}), ..., d_\phi(y^{(m)})\}$
    Update encoder q_λ and decoder p_θ by (14)
  end for
  for d-steps **do**
    Use current p_θ to generate negative examples f' and combine with given positive examples x' from $\mathcal{S}$
    Train discriminator d_ϕ via minimizing the cross entropy
  end for
  Update decoder-copy p_β by (4)
until SeqVAE converges

All in all, Algorithm 1 describes the full details of the training process of SeqVAE using (14).

4 Experiments

4.1 Synthetic data experiments

To test the efficacy and add understanding of SeqVAE, we compare it with plain VAE and SeqGAN. To facilitate comparison with SeqGAN, we conduct a simulated test with synthetic data [28]. Firstly, a neural network (for example, LSTM [13]) that can generate sequence data is constructed, call it $oracle$, and initialize it randomly. Next, use $oracle$ to generate the training set, and use the training set to train the models we want to compare. Finally, use $oracle$ to evaluate the effect of the trained models, because $oracle$ implicitly defines the distribution of the training set. So, a perfect evaluation metric (Negative log-likelihood, NLL) should be:

$$\mathrm{NLL}_{\mathrm{oracle}} = -\mathbb{E}_{Y_{1:T}\sim G_\theta}\left[\sum_{t=1}^{T} \log G_{\mathrm{oracle}}\left(y_t|Y_{1:t-1}\right)\right] \quad (15)$$

where G_θ represents the models we want to compare and G_{oracle} represents $oracle$.

4.1.1 Training setting

For $oracle$, we use the standard normal distribution $\mathcal{N}(0, 1)$ to initialize its parameters.

Then use $oracle$ to generate a training set $\mathcal{S}$ containing 10,000 sequences.

In the experiment[1], we compare the generation effects of plain VAE, SeqGAN, SeqVAE_1, and SeqVAE_2. SeqVAE_1 and SeqVAE_2 represent the calculation of the policy-gradient loss using (8) and (14), respectively. The detailed settings about them are described below.

Both plain VAE, SeqVAE_1 and SeqVAE_2 contain encoder and decoder. In order to control variables, we all use the same structure and hyperparameter settings. We use a two-layer bidirectional LSTM for the encoder, and there is a residual connection [12] between different layers. We use a single-layer unidirectional LSTM for the decoder. In addition, we use the same learning rate l and eventually use the Adam algorithm to update the parameters. In SeqVAE_1 and SeqVAE_2, we need to set the discriminator. Some deep neural networks (DNN) [26] can be used as discriminator. In order to control variables, we use the same CNN [16]. The discriminator has its own learning rate, and we also use the Adam algorithm to update its parameters.

4.1.2 Result

Table 1 illustrates the comparison of various sequence generation models on NLL_{oracle}. In the generation of sequences of various lengths, both SeqVAE_1 and SeqVAE_2 are far superior to plain VAE, which shows that SeqVAE with the policy-gradient loss does effectively reduce the deviation of vae's optimization goal. In addition, we can see that SeqVAE_1 gets better and better as the length of the sequence increases and it gradually exceeds SeqGAN, although SeqVAE_1 adds the policy-gradient loss in a way that is almost identical to SeqGAN. And, it can

[1] https://github.com/INotWant/seqvae

Table 1 Comparison of various sequence generation models on NLLoracle (best results presented in bold font)

Sequence length	Plain VAE	SeqGAN	SeqVAE_1	SeqVAE_2
8	8.53	7.54	8.19	**7.46**
20	8.79	8.49	8.53	**8.42**
64	8.40	8.32	**8.18**	8.19

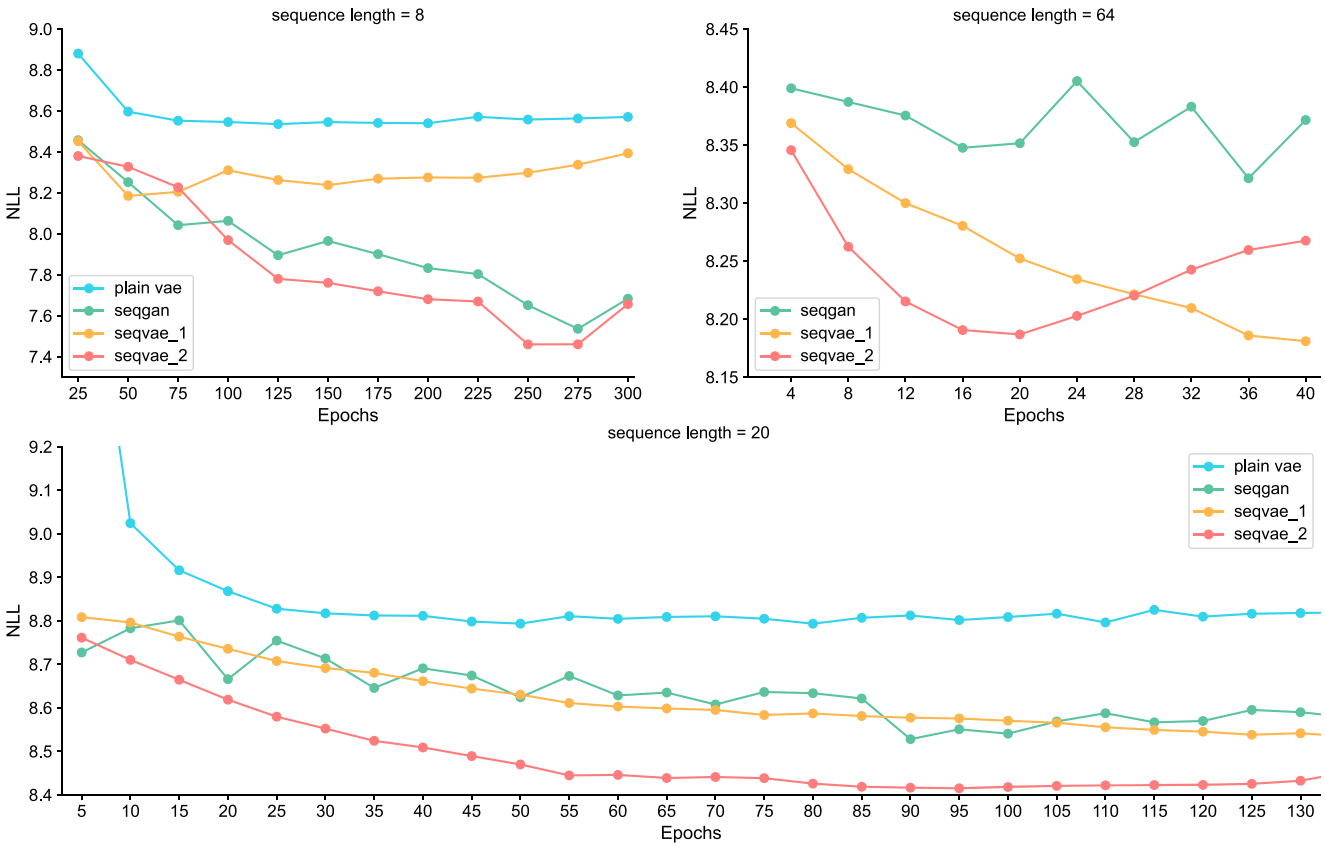

Fig. 2 Negative log-likelihood convergence w.r.t. the training epochs

Table 2 Music melody generation performance comparison (best results presented in bold font)

Model	SPB	QN		
		4	6	8
Nottingham	*0.37*	*67.17%*	*46.65%*	*45.52%*
plain VAE	0.33	75.37%	55.44%	54.25%
MusicVAE	0.30	74.38%	51.86%	50.61%
SeqVAE_1	**0.37**	**68.42%**	**47.71%**	**46.51%**
SeqVAE_2	**0.37**	70.63%	48.61%	47.26%

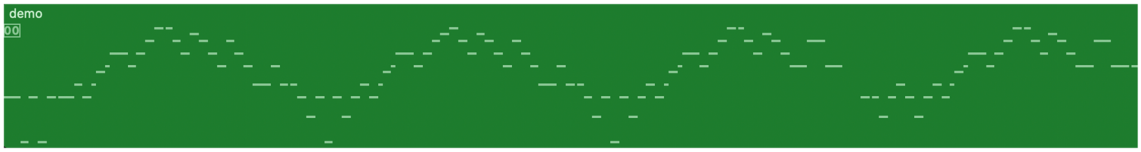

Fig. 3 The melody generated by SeqVAE_1, the abscissa represents time, and the ordinate represents pitch

be seen from the learning curve Fig. 2 that the training process of SeqVAE_1 is more stable than SeqGAN, which benefits from the KL-divergence loss. Therefore, we believe that SeqVAE can be used as a solution to improve the training stability of GAN. In almost all cases, SeqVAE_2 is superior to other models, which proves the effectiveness of our proposed method of calculating the policy-gradient loss. In the process of calculating the policy-gradient loss of SeqGAN, reward needs to be obtained for each time step. In the process of calculating the policy-gradient loss of SeqVAE_2, there is no need to obtain reward for each time step, only the reward for the entire time series is calculated, and there is no approximation of reward. We think that SeqVAE_2's approach can enhance the learning of fragments in the sequence.

And, SeqVAE has other advantages over SeqGAN. SeqVAE can use latent vector to control generation and supports interpolation, it is less prone to generate duplicate data than SeqGAN, and it has better interpretability and easier training.

4.2 Music melody generation

In order to explore the practical application of SeqVAE, we compare various sequence generation models by generating music melody. In the experiment, our task is to generate 4/4 beat melodies with a fixed length of 16 bars. Therefore, the length of the generated sequence is 256, which is much longer than ordinary text sentences.

In the experiments, we use Nottingham[2] dataset. After preprocessing, 2488 melodies with a fixed length of 16 bars are obtained and we use these melodies to form the training set. The way we encode our melodies is borrowed from magenta[3]. About the evaluation metric, we initially planned to use BLEU [22]. However, during the experiment, it was found that the BLEU value cannot effectively evaluate the quality of the generated melody. The melody with poor rhythm can get higher BLEU value. Inspired by the work of [7], we evaluate the model by comparing the statistics on the generated sample set with the training set. The closer the statistics on the training set, the better the performance of the model. The statistics we use are SPB and QN, which represent the music span of each measure and the percentage of "qualified" notes to the total number of notes (a "qualified" note means a note that exceeds the set duration).

Table 2 shows the comparison of various sequence generation models on the task of melody generation. We set 3 duration criteria for the QN , which are 4 times, 6 times and 8 times the length of the sixteenth note, respectively. The second row shows the values of the Nottingham dataset under these statistics. Comparing this row with other rows, we can see that SeqVAE (regardless of SeqVAE_1, SeqVAE_2) is much better than plain VAE, which shows that SeqVAE does reduce the deviation of vae's optimization goal. In addition, SeqVAE is better than MusicVAE, which to a certain extent shows that SeqVAE alleviates the "post-collapse" problem in training. Figure 3 shows a melody generated by SeqVAE_1. From this we can see its rhythm and movement are in accordance with the music theory.

5 Conclusion

In the paper, we propose a variant of recurrent VAE, called SeqVAE, which introduces policy gradient to VAE to reduce the deviation of ELBO. In SeqVAE, the original encoder is maintained, another original decoder is regarded as a policy function, a decoder is added for Monte Carlo search, and a new discriminator is used to obtain the reward signal for reinforcement learning. In our synthetic data experiments, SeqVAE_2 surpassed all baselines and achieved excellent performance. In the experiment of music melody generation, SeqVAE has been demonstrated its advantages in generating sequences with long-term structure and can alleviate the "post-collapse" problem. In future work, we plan to study whether our proposed method of calculating the policy-gradient loss can play a role in GAN.

Acknowledgements Finally, I must express my very profound gratitude to my parents and to my thesis advisor for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

[2] https://github.com/jukedeck/nottingham-dataset

[3] https://github.com/tensorflow/magenta

References

1. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan. arXiv:1701.07875
2. Bachman P, Precup D Data generation as sequential decision making. In: Advances in Neural Information Processing Systems, pp. 3249–3257
3. Bao J, Chen D, Wen F, Li H, Hua G Cvae-gan: fine-grained image generation through asymmetric training. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2745–2754
4. Bengio S, Vinyals O, Jaitly N, Shazeer N Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1171–1179
5. Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R, Bengio S (2015) Generating sentences from a continuous space. arXiv:1511.06349
6. Carter S, Nielsen M (2017) Using artificial intelligence to augment human intelligence. Distill 2(12):e9
7. Dong HW, Hsiao WY, Yang LC, Yang YH Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: Thirty-Second AAAI Conference on Artificial Intelligence
8. Engel J, Resnick C, Roberts A, Dieleman S, Norouzi M, Eck D, Simonyan K Neural audio synthesis of musical notes with wavenet autoencoders. In: Proceedings of the 34th International Conference on Machine Learning, vol 70, pp 1068–1077. JMLR. org
9. Goodfellow I (2016) Generative adversarial networks for text http://goo.gl/wg9DR7
10. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680
11. Ha D, Eck D (2017) A neural representation of sketch drawings. arXiv:1704.03477
12. He K, Zhang X, Ren S, Sun J Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
13. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
14. Huszár F (2015) How (not) to train your generative model: Scheduled sampling, likelihood, adversary. arXiv:1511.05101
15. Karras T, Laine S, Aila T A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4401–4410
16. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv:1408.5882
17. KingmaD A (2015) A methodforstochasticoptimization. arxiv: 1412.6980
18. Konda VR, Tsitsiklis JN Actor-critic algorithms. In: Advances in neural information processing systems, pp 1008–1014
19. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. arXiv:1312.5602
20. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
21. Oord A. v. d, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) Wavenet: A generative model for raw audio. arXiv:1609.03499
22. Papineni K, Roukos S, Ward T, Zhu WJ Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, pp 311–318. Association for Computational Linguistics
23. Roberts A, Engel J, Raffel C, Hawthorne C, Eck D (2018) A hierarchical latent vector model for learning long-term structure in music. arXiv:1803.05428
24. Semeniuta S, Severyn A, Barth E (2017) A hybrid convolutional variational autoencoder for text generation. arXiv:1702.02390
25. Sutton RS, McAllester DA, Singh SP, Mansour Y Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems, pp. 1057–1063
26. Veselý K, Ghoshal A, Burget L, Povey D Sequence-discriminative training of deep neural networks. In: Interspeech, vol 2013, pp 2345–2349
27. Wang H, Qin Z, Wan T Text generation based on generative adversarial nets with latent variables. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 92–103
28. Yu L, Zhang W, Wang J, Yu Y Seqgan: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence
29. Zhou F, Yang S, Fujita H, Chen D, Wen C (2020) Deep learning fault diagnosis method based on global optimization gan for unbalanced data. Knowl-Based Syst 187(104):837
30. Zhu JY, Park T, Isola P, Efros AA Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp. 2223–2232