MuseBar: Alleviating Posterior Collapse in Recurrent VAEs Toward Music Generation

Huiyao Wu and Maryam Tavakol(✉)

Eindhoven University of Technology, Eindhoven, The Netherlands
h.wu1@student.tue.nl, m.tavakol@tue.nl

Abstract. Machine learning has shown remarkable artistic values and commercial potentials in the music industry. Recurrent variational autoencoders (RVAEs) have been widely applied to this area due to the condensing, inclusive, and smooth nature of their latent space. However, RNNs are powerful auto-regressive models on their own, where the decoder in a RVAE can be strong enough to work independently from the encoder. When this happens, the model degrades from an autoencoder to a traditional RNN, which is known as *posterior collapse*. In this paper, we propose a cost-effective bar-wise regulation schema called *MuseBar* to alleviate this problem for music generation. We impose a prior on the hidden state of every music bar in the RNN encoder, instead of only on the last hidden state as in the standard RVAEs, such that the latent code is learned under stronger regulations. We further evaluate our proposed method, quantitatively and qualitatively, with extensive experiments on manually scraped musical data. The results demonstrate that the bar-wise regulation significantly improves the quality of the latent space in terms of Mutual Information and Kullback-Leibler divergence.

Keywords: Music generation · Variational autoencoder · Recurrent neural networks · Posterior collapse

1 Introduction

Recent advances in Artificial Intelligence (AI) have exhibited great values in creative arts such as music composing [18], poem writing [13], painting imitation [5], and so on. Creating arts using AI techniques is efficient, imaginative, inspiring, and for music generation, it can lead to additional commercial benefits. Because of AI, people now have extensive exposure to sophisticated yet user-friendly creation, remixing, and learning tools e.g., Magenta, Flow Machines, MuseNet, etc. Moreover, it can be used for therapeutic purpose as certain types of music have been proven effective in suppressing beta and gamma rhythms in the brain that are correlated with depression and anxiety [9]. Thus, pre-defining various output patterns via AI technologies can undoubtedly save a lot of human labor and produce more effective therapeutic music.

T. Bouadi et al. (Eds.): IDA 2022, LNCS 13205, pp. 365–377, 2022.
https://doi.org/10.1007/978-3-031-01333-1_29

However, successfully modeling the long sequences of notes in the musical data is very challenging. On the one hand, convolutional neural networks (CNNs) have been adopted to capture the patterns in the musical data [2,11]. CNNs are used to distill the musical features, in which, music data is treated as consecutive images and each image represents a bar in the music notation. These methods are utilized in polyphonic music modeling as they can capture the common patterns among multiple tracks. Nonetheless, they fail to generate the long- and short-term structure of the music, which can not be ignored in long music sequences as the transitions among the bars is what brings the tuneful melodies.

On the other hands, deep generative models [7] have been widely applied for music generation. Recurrent variational autoencoders (RVAEs) are among the most popular frameworks for this purpose due to the representative and continuous nature of their latent space. Nevertheless, RNNs themselves are typically used on their own as powerful auto-regressive models of sequences. The decoder in a recurrent VAE is sufficiently capable of modeling the sequential data and might ignore the latent code from the encoder. With the latent code disregarded, the model degrades from an autoencoder to a traditional RNN. This is known as *posterior collapse*, or *KL vanishing* [1], which could get worse for music generation as music notes are commonly of longer sequences. To alleviate this problem, most of the existing approaches mainly focus on either designing a stronger encoder by stacking LSTM units [16,17] or restricting the power of decoder by introducing additional modules [14]. However, both approaches lead to bulky models with significantly more parameters to learn.

In this paper, we introduce a light-weighted bar-wise regularization technique to address the posterior collapse in RVAEs without incorporating extra parameters and/or additional modules. In a standard RVAE, a prior based on a standard Gaussian distribution is imposed only on the last hidden state of the RNN encoder, which might not be enough for long sequences, and can result in an inaccurate representation of the latent code. Therefore, inspired by the ideas from Li et al. [10], we propose MuseBar, a bar-wise regulation scheme to effectively compress the data into the latent space. Subsequently, the Gaussian prior is applied on multiple hidden states of the RNN-based encoder, and in this way, a stronger regulation is imposed on the model and will theoretically produce a more informative latent space, in particular, during the early phases. Empirical study on manually scraped musical data from different genres shows that MuseBar can effectively mitigate posterior collapse and outperforms certain baselines in terms of the quality of the latent space as well as the overall performance.

2 Background

2.1 MIDI Representation

In order to make music notes accessible to the computer, they have to be encoded according to a certain unified grammar. Musical Instrument Digital Interface (MIDI) is one of the most commonly used formats which we adapt in this paper. MIDI is a technically standard format to describe a protocol, a digital interface,

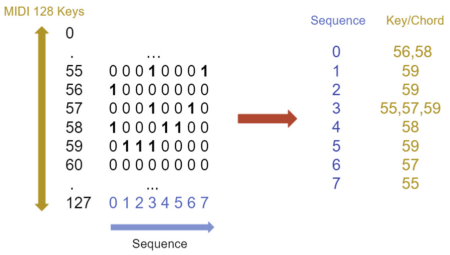


Fig. 1. Comparison of piano-roll matrix and MIDI notes.

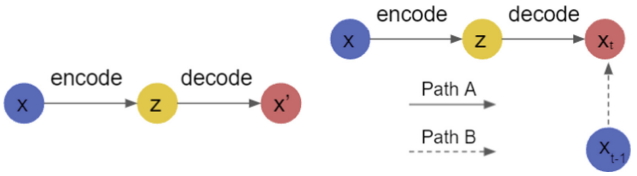
and connectors, which makes the interaction between various electronic musical instruments, software, and devices possible [15].

Music notes in the MIDI format are often processed to the piano-roll representation, which can be further translated into a matrix indicating which note(s) at a certain pitch is played, at what velocity, and at which time frame. Simply put, a piano-roll matrix is a mathematical representation of a MIDI file that only focuses on musical notes and their key attributes. As depicted in Fig. 1, the pitch value in MIDI files usually ranges from 0 to 127, hence in total, we have 128 different pitches. The **1**s in the figure represent the note(s) with particular pitch indicating when the left bar is being played. Although MIDI representation is not able to distinguish whether a key is held for multiple time steps or consecutively pressed during a certain time frame, it is still a practical format for digital composition in music generation due to its adaptability.

2.2 Generative Models

Autoencoders. An autoencoder (AE) is a neural network that aims at successfully replicating its input $\boldsymbol{x}$ to output $\boldsymbol{x}'$. There are two key parts in an AE, an encoder that compresses the input signal into the latent code $\boldsymbol{z}$, and a decoder that reconstructs the input data from $\boldsymbol{z}$. AEs have been widely used for dimensionality reduction and feature extraction. However, the latent code from an AE is a discrete vector of a fixed length, making it difficult to interpolate. When sampling from such latent vector, we may end up with unrealistic output as the decoder has never encountered some parts of the input before.

Variational Autoencoders. Compared to AEs, the latent space of variational autoencoders (VAEs) is designed to be continuous and thus allows for random interpolation and sampling. This is achieved by casting the input data to a distribution instead of a latent code of fixed length. A VAE [8] samples the latent vector $\boldsymbol{z}$ from a prior distribution $p(\boldsymbol{z})$, which is controlled by parameters μ and σ. The encoder is then denoted in conditional probability as $Q_\phi(\mathbf{z} \mid \mathbf{x}_i)$, where ϕ indicates the weights of the encoder network. Correspondingly, a decoder is represented as $P_{\boldsymbol{\theta}}(\mathbf{x}'_i \mid \mathbf{z})$, with θ being the weights of the decoder.



(a) CNN-based VAE (b) RNN-based VAE

Fig. 2. Illustration of information flow in VAEs.

The backbone of both the encoder and decoder in a VAE can be of various neural network structures, while we only focus on recurrent neural networks (RNNs) in this paper, as musical notes are sequential data in essence. More specifically, the encoder Q_ϕ is composed of an LSTM module that outputs a series of hidden states $h_1, h_2, \ldots, h_T$ for a given input sequence $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$. The hidden states are used to generate μ and σ, which are the parameters of the distribution over the latent code $\boldsymbol{z}$. The latent vector $\boldsymbol{z}$ is sampled from this distribution and is then utilized to initialize the states of the decoder $P_{\boldsymbol{\theta}}$. The decoder network is also composed of an LSTM module and can auto-regressively reconstruct the input sequence to output $\boldsymbol{x}' = \{x'_1, x'_2, \ldots, x'_T\}$. The overall model is trained to both learn an approximate posterior $Q_\phi(\boldsymbol{z} \mid \mathbf{x})$ which is close to the prior $P(\boldsymbol{z})$, as in a standard VAE and reconstruct the input sequence (i.e., $x'_i = x_i, i \in \{1, \ldots, T\}$). The loss function is subsequently defined as

$$\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i^{1:t}\right) = \mathbb{E}_{Q_\phi\left(\mathbf{z}^T \mid \mathbf{x}_i^{1:t}\right)}\left[\log P_{\boldsymbol{\theta}}\left(\mathbf{x}_i^{1:t} \mid \mathbf{z}^T\right)\right] - D_{\mathrm{KL}}\left(Q_{\boldsymbol{\phi}}\left(\mathbf{z}^T \mid \mathbf{x}_i^{1:t}\right) \| P\left(\mathbf{z}^T\right)\right), \tag{1}$$

where $\mathbf{x}_i^{1:t}$ represents the sequential input data and $\mathbf{z}^T$ represents the latent code sampled from the last hidden state of the encoder.

Posterior Collapse. Posterior collapse, also known as *KL vanishing*, is a notoriously difficult problem in variational autoencoders, which makes it hard to train an effective model due to the vanish of the KL term in Eq. (1). This phenomenon is particularly obstinate when an RNN is exerted as the backbone of VAE, since the temporal receptive field in RNNs can be unlimited in essence.

An intuitive explanation of such phenomenon is displayed in Fig. 2 [4]. For a standard VAE with a CNN as the backbone, Fig. 2(a) shows that there is only one path moving from encoder Q_ϕ, to the latent code $\boldsymbol{z}$, and then to the decoder $P_{\boldsymbol{\theta}}$, when reconstructing the input data $\boldsymbol{x}$. Such a uni-directional flow limits the leak of information along the way and thus makes it easier to collaboratively train the model. However, we will end up with an additional information flow brought by the auto-regressive RNN if the traditional CNN decoder is replaced by an RNN module. For an RNN decoder, the inputs come from two paths, as shown in Fig. 2(b), one is from the latent code $\boldsymbol{z}$ and the other one is from the output of the previous time step. Similar to the standard VAEs, $\boldsymbol{z}$ serves

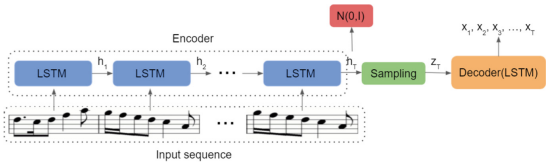


Fig. 3. Standard recurrent VAE. (Color figure online)

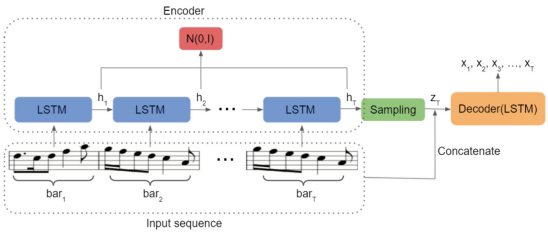


Fig. 4. Schematic of the bar-wise regulation on recurrent VAE.

as the global feature vector that supervises the replication of $\boldsymbol{x}$, whereas parts of the ground-truth information of $\boldsymbol{x}$ suffers from leaking at every time step of the sequential decoding. As a result, the decoder becomes powerful enough to generate data on its own and could not utilize stochastic information brought by the trained latent space.

3 MuseBar

Effectively compressing the data to the latent space is crucial for the decoder training, as the latent code is later used to initialize and supervise the reconstruction [1,4,6]. Nevertheless, to train an informative latent space especially at the early stages is very challenging, as the encoder has not yet learned to condense the input data. The latent code is hence not strong enough to guide the decoder given its auto-regressive nature. Therefore, the decoder fails to cooperate with the encoder and the latent code becomes meaningless. In order to penalize the decoder from working independently, standard RVAEs, also illustrated in Fig. 3, add one regularizer (red box) into the bottleneck layer to force the latent space to follow the standard Gaussian distribution. However, only a single regulation on the last hidden state of the encoder is insufficient to guarantee an optimal latent space, especially when capturing the long-term dependencies.

Consequentially, following the light of step-wise regulation [10], we propose MuseBar, a bar-wise regularizer, as displayed in Fig. 4, to impose stronger regulations on the hidden states generated from multiple bars in a sequence of music. The resulted Evidence Lower BOund (ELBO) loss is hence

$$\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i\right)_{\mathrm{b}} = E_{Q_\phi(\mathbf{z}^{\mathrm{T}}|\mathbf{x}_i)}\left[\log P_\theta\left(\mathbf{x}_i \mid \mathbf{z}^T\right)\right] - \frac{1}{b}\sum_{t=1}^{b} D_{\mathrm{KL}}\left(Q_\phi\left(\mathbf{z}^t \mid \mathbf{x}_i^{1:t}\right) \| P\left(\mathbf{z}^t\right)\right), \tag{2}$$

where the final KL term is obtained from the average divergence of each bar in the input sequence.

Given a musical sequence of length L with a number of b bars, a recurrent encoder compresses the input into a low-dimensional latent space where only most representative information is retained. Subsequently, the latent code $\boldsymbol{z}$ of length l is sampled from this space to initialize the decoder, which will be used to reconstruct the input sequence. The entire VAE network is trained under the supervision of a reconstruction loss imposed on the input and output, which in our case, is a binary cross-entropy loss, and a regularizer on the latent space and standard Gaussian distribution, which is quantified as Kullback-Leibler divergence. MuseBar adds a Gaussian distribution prior to every bar of the input sequence as illustrated in Fig. 4. We choose bars as regulation units rather than single beats because musical sequences are usually at least 20 times longer than texts, hence, adding regulation on every beat can lead to a significant increase in the computational costs. Apart from the bar-wise regulation, latent code $\boldsymbol{z}$ is concatenated with the input sequence to form the final input for the decoder instead of solely being used for initialization. This way, more information is passed on to the decoder and theoretically brings richer context.

The main structure of the model consists of a 2-layer LSTM for both encoder and decoder with hidden units of size 512 and input/output dimension of 128, which is the number of pitches in the MIDI representation. Furthermore, the vector $\boldsymbol{z}$ is sampled from the latent space and is both used to initialize the decoder and concatenated with every sequence as the final input for the decoder. In the output layer of the decoder, a sigmoid function is applied in order to convert the output into probabilities. Finally, a binary construction (BCE) loss is used along with the KL divergence to train the model in an end-to-end manner.

4 Empirical Study

In this section, we conduct experiments to evaluate the performance of MuseBar compared to several baselines for music generation. The music data is collected from Freepianotutorials[1] and Lakh MIDI Dataset[2], consisting of five major genres which include pop, rock, classical, jazz, and electronic. For each genre, we collect 100–300 songs, depending on the availability of monophonic MIDI files and the productivity of the corresponding musicians. Every song is further preprocessed to be of the same length by either repetition or interruption and then concatenated into a long piece. For simplicity, all musical data is rendered by piano and collected in monophonic MIDI format, with the help of **Music21** library, which is a Python-based toolkit for computer-aided musicology.

[1] https://www.freepianotutorials.net.
[2] https://colinraffel.com/projects/lmd.

In our setting, the batch size equals to the number of musicians/genres so that we can pick one specific genre/musician based on the batch index later on. Models with different parameter settings are trained for 3k epochs and learning rate is set to 0.001 as an upper limit for the Adam optimizer. During training, we randomly select a snippet of 240 time steps from each musician/genre to feed the model, as 240 is the average time steps needed for a basic structure in a song (intro, verse, pre-chorus, chorus, and bridge) [12], which is approximately 30 s. In addition, we take 80 bars in one snippet on a 4/4 time signature, each contains 4 time steps. To evaluate the performance of our model, we conduct the following experiments.

Overall Performance. To verify the effectiveness of the bar-wise regulation, we first compare MuseBar to two baselines: vanilla VAE (**LSTM-VAE**) [3], which is simply composed of an LSTM unit without any special training strategy or regulations, and the one proposed by Bowman et al. [1] (**VAE-BOW**), which is trained under a weight annealing strategy. In the latter, the importance of KL term is progressively raised over time in order to force the model to first learn the latent code and then utilize it. Additionally, we train the bar-wise regularized RVAE combined with the weight annealing method (**MuseBar-BOW**).

Music Genres. Monophonic music of different genres vary drastically in terms of the transition intensity and pitch ranges. Classical music tends to have more frequent transitions and wider range of pitch class than pop or rock music, which is the reason why classical music sounds richer in texture and melody. With the aim of exploring the impact of different genres, we train the regularized recurrent VAE with the same network structure on datasets of 5 different genres, respectively pop, rock, classical, jazz and electronic.

Hyperparameter Exploration. Autoencoders, or rather the encoder component of them, in general are compression algorithms. Therefore, the size of the latent space greatly impacts the effectiveness and efficiency of the model. A latent code with a small size might not properly capture all the information needed to reconstruct the input, while a larger latent space might end up with too many dead units which consequently become more costly to train. Thus, we tune the model to identify the optimal length of the latent vector $\boldsymbol{z}$ as a hyper-parameter and we vary the length of the latent code from 8 to 128 in a geometric sequence in this experiment. Furthermore, we explore the number of regularizers as another hyper-parameter of the model. To this end, with the input size of 240 time steps consisting of 60 bars under the time signature of 4/4, we experiment different numbers of regularizers from {5, 10, 15, 30, and 60} on both full dataset and separate genres, respectively, corresponding to regulation on every 12, 6, 4, 2, and 1 bar(s).

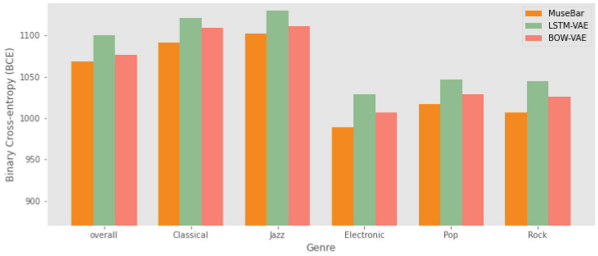


Fig. 5. Comparison of Binary Cross-entropy (BCE) in MuseBar and baselines.

4.1 Evaluation Metrics

In general, VAEs use the Evidence Lower Bound (ELBO) measure to estimate the negative log likelihood of the data points under the learned distribution to assess the reconstruction loss. Hence, ELBO can be used as the quantitative measure of the overall performance of the model. We further investigate the binary cross-entropy (BCE) and Kullback-Leibler divergence (KL), which are two sub-components of ELBO, in our evaluation.

In addition to the inherent KL term, one of the most commonly used quantitative method for evaluating the quality of encoding is called Mutual Information (MI). In information theory, mutual information is a measure of mutual dependence of two given random variables, which can also be considered as the reduction in the uncertainty about one random variable given the knowledge of another. As such, we compute the MI between the latent space and the sampled latent code to get an idea of how deductive the latent space is from which a latent variable is sampled. The difference between MI and KL is that MI indicates how much information is encoded in the latent code from the latent space, while KL determines how far the latent space is from the standard Gaussian distribution.

4.2 Performance Results

Overall Performance. The general performance of different methods is evaluated on the joint test set of all genres with the input size of 240×128 and the latent vector of length 100. The regulation is imposed on every two bars as this leads to the best performance according to the exploration study conducted later on. In order to reduce the effect of randomness, the results are averaged over 10 runs. The performance is further compared with vanilla recurrent VAE (LSTM-VAE) and VAE trained with weight annealing strategy (VAE-BOW), which are shown in Fig. 5. In this figure, only the binary cross-entropy loss is displayed as the Kullback-Leibler divergence and mutual information (MI) are trivial (close to zero) in LSTM-VAEs compared to MuseBar.

In addition, Table 1 summarizes the evaluation results of all methods in terms of three metrics on the entire data as well as each genre. For the entire data (the Overall row), according to binary cross-entropy (BCE), Kullback-Leibler divergence (KL), and Mutual Information (MI), the obtained results illustrate that

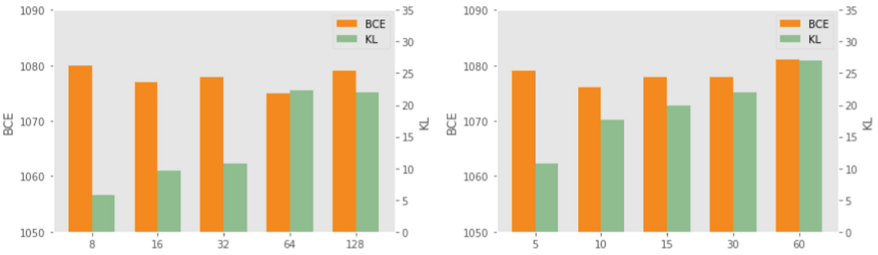
Table 1. The performance of all methods on the collected dataset.

		LSTM-VAE	VAE-BOW	MuseBar	MuseBar-BOW
Overall	BCE	1100	1076	1078	**1068**
	KL	0.1	19	22	**27**
	MI	0.7	17	20	**21**
Classical	BCE	1121	1109	1111	**1091**
	KL	0.08	17	20	**30**
	MI	0.5	10	11	**20.3**
Jazz	BCE	1130	1111	1113	**1102**
	KL	–	14	21	**32**
	MI	–	15	20	**21.7**
Electronic	BCE	1029	1007	1001	**989**
	KL	0.1	23	29	**35.7**
	MI	0.9	20	21	**24**
Pop	BCE	1047	1029	1021	**1017**
	KL	0.09	16	27	**32.9**
	MI	0.07	15	21	**23.4**
Rock	BCE	1045	1026	1019	**1007**
	KL	0.08	21	27	**29**
	MI	0.03	20	23	**24**

both VAE-BOW and MuseBar enhance the vanilla recurrent VAE, with MuseBar slightly better than VAE-BOW, but the best performance is achieved when two strategies are combined, i.e., training MuseBar with weight annealing method. Note that the weight annealing strategy balances the two terms in ELBO, which is beneficial when the input size and latent dimension vary drastically. Samples of music generated from our model can be found online[3].

Music Genres. In addition to training on the complete dataset, we further investigate the performance of bar-wise regulation and corresponding baselines on each genre used in this paper. The results from both Fig. 5 and Table 1 demonstrate that different genres react differently to the regularization. For classical and jazz, VAE-BOW slightly outperforms MuseBar in terms of BCE, while for other genres MuseBar achieves a better performance. This happens due to both the specialty of the strategy and attributes of the music itself. VAE-BOW is designed to balance the two terms in ELBO, which accordingly focuses on global adjustment, unlike bar-wise regularizer, which is targeted at local bar-wise texture. Therefore, MuseBar is more suitable for genres that are less variant such as electronic and rock. Not Surprisingly, electronic is more sensitive to any form of

[3] Link to the sample musics.



(a) Length of latent code z (b) Number of regularizers

Fig. 6. BCE loss and KL-divergence with different parameter settings. (Color figure online)

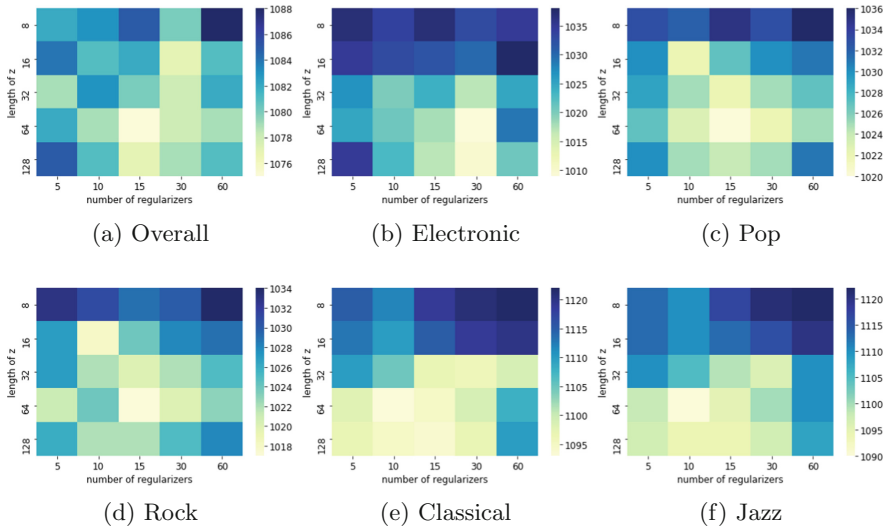
optimization compared to other genres. We believe this is because of the repetitive property of electronic music, which makes it easier to model and predict. On the other hands, pop and rock music react similarly to either kind of enhancements. We reckon this is because of the indistinct boundary between these two genres. For example, Coldplay can be considered as a rock band and a pop band at the same time. Same applies to the Beatles, Greenday, One Direction, etc. Needless to say that this is also biased toward the collected data.

Hyperparameter Exploration. In this experiment, we first explore the impact of different lengths of the latent code. Experiments in this part are conducted on the complete dataset with 30 regularizers. Figure 6(a) plots the performance of MuseBar in terms of varying BCE (orange) and KL (green) measures.

As shown in Fig. 6(a), the size of the latent space does not necessarily have a big influence on BCE as it does on KL. The BCE remains stable with slight fluctuation from 1075 to 1080, while KL varies from 5.7 to 22.3. However, a larger latent space (128) does not outperform a smaller one (64) in terms of both KL and BCE, which means that our model does not require a very large latent space to represent the compression of the input data. In the above experiments, we adapt the size of 100 as the optimal latent dimension for the complete dataset, aiming to effectively and efficiently compress the input sequence.

Additionally, we aim to verify the influence of the number of regularizers on the model. Hence, we evaluate the performance of MuseBar with 5, 10, 15, 30, and 60 regularizers on both full dataset and separate genres, which respectively correspond to regulation on every 12, 6, 4, 2, and 1 bar(s). In this experiment, the latent code is fixed to the length of 100. The results are displayed in Fig. 6(b) for BCE (orange) and KL (green) terms.

Similarly, Fig. 6(b) demonstrates that the number of regularizers does not necessarily have a great influence on BCE as it does on KL divergence. The BCE remains stable with slight fluctuation from 1076 to 1081, while KL term varies from 10.7 to 27.3. The obtained results verify that a stronger regulation leads to a higher KL divergence as the KL reaches its maximum when 60 regularizers



(a) Overall (b) Electronic (c) Pop

(d) Rock (e) Classical (f) Jazz

Fig. 7. Heatmap of BCE with variation of length of z and number of regularizers.

are added to the model. Nevertheless, more regularizers (60) does not surpass less regulations (30) in terms of BCE, which indicates that vigorous supervision is not necessarily beneficial to reconstruction. We believe this is sensitive to the music genre. Genres with more flexible and creative patterns might need more regulations while genres with less variations such as electronic does not require strong regulations.

Consequently, we further investigate the interaction of the number of regularizers and the length of the latent code on each genre. To this end, we conduct extensive exploration experiments, where the achieved results are demonstrated as heatmaps in Fig. 7. We employ BCE as the evaluation metric for this set of experiments as the reconstruction loss is the core measure of interest to evaluate the overall performance of the method.

Overall, Fig. 7 shows that BCE is negatively correlated to the number of regularizers as well as the length of the latent code. When the latent code is of length 8, all music genres reach the largest BCE as the size of the latent space is not sufficient enough to represent the input sequence. However, there is no significant difference regarding BCE when the latent code is above 64. This also testifies the aforementioned assumption that the latent space of a large size might be redundant as it might include inactivated units, which is not utilized by the decoder at all. For various genres, the optimal combination of the number of regularizers and the size of the latent code is also not the same. For example, electronic in general needs more regulations compared to classical or jazz music to reach an optimal BCE. We believe this is due to the repetitive/predictive music notes progression of the electronic genre. Moreover, we conclude that it is easier to reconstruct an electronic piece than a classical one according to the range of the BCE indicated with the side bar.

5 Conclusions

In this paper, we aimed to mitigate the posterior collapse problem in recurrent VAEs in the context of music generation. Inspired by the step-wise VAE [10], we proposed MuseBar, which adds stronger regulations to the hidden states of the recurrent encoder during the training process, ensuring a more informative latent space that can be used as a global guidance for the auto-regressive decoder. Extensive experiments are conducted to analyze the impact of the length of the latent code, number of regularizers, and various music genres on the performance. Our model by itself as well as combined with weight annealing strategy (MuseBar-BOW) significantly outperforms vanilla recurrent VAE (LSTM-VAE) and VAE trained with weight annealing strategy (VAE-BOW) in terms of binary cross-entropy, Kullback-Leibler divergence, and Mutual Information. Although the bar-wise regulation effectively enhances the overall performance of RVAEs and mitigates the KL vanishing problem, there are some limitations that hinder the further improvements, such as the MIDI representation and the LSTM architecture. To address the aforementioned issues, we aim to seek for a more fidelitous but also cost-efficient digital representation, and exploit more sophisticated architectures such as bi-directional or pyramid LSTM encoder, that are left for the future work.

References

1. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S.: Generating sentences from a continuous space. In: Proceedings of the Twentieth Conference on Computational Natural Language Learning (2015)
2. Dong, H.W., Hsiao, W.Y., Yang, L.C., Yang, Y.H.: MuseGAN: multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, vol. 32, no. 1 (2018)
3. Fabius, O., Van Amersfoort, J.R.: Variational recurrent auto-encoders. arXiv preprint arXiv:1412.6581 (2014)
4. Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., Carin, L.: Cyclical annealing schedule: a simple approach to mitigating KL vanishing. In: Proceedings of NAACL (2019)
5. Ha, D., Eck, D.: A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477 (2017)
6. Huang, A., Wu, R.: Deep learning for music. arXiv:1606.04930v1 (2016)
7. Jiang, J., Xia, G.G., Carlton, D.B., Anderson, C.N., Miyakawa, R.H.: Transformer VAE: a hierarchical model for structure-aware and interpretable music representation learning, pp. 516–520 (2020)
8. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: Second International Conference on Learning Representations (2013)
9. Kirk, R., Abbotson, M., Abbotson, R., Hunt, A., Cleaton, A.: Computer music in the service of music therapy: the MIDIGRID and MIDICREATOR systems. Med. Eng. Phys. **16**(3), 253–258 (1994)
10. Li, R., Li, X., Chen, G., Lin, C.: Improving variational autoencoder for text modelling with timestep-wise regularisation. arXiv preprint arXiv:2011.01136 (2020)

11. Malekzadeh, S., Samami, M.: Classical music generation in distinct dastgahs with alimnet ACGAN. arXiv preprint arXiv:1901.04696 (2019)
12. McIntyre, P.: Creativity and cultural production: a study of contemporary western popular music songwriting. Creat. Res. J. **20**(1), 40–52 (2008)
13. Oliveira, H.G., Hervás, R., Díaz, A., Gervás, P.: Adapting a generic platform for poetry generation to produce Spanish poems, pp. 63–71 (2014)
14. Roberts, A., Engel, J., Raffel, C., Hawthorne, C., Eck, D.: A hierarchical latent vector model for learning long-term structure in music, pp. 4364–4373 (2018)
15. Rothstein, J.: MIDI: A Comprehensive Introduction, 7th edn. A-R Editions, Middleton (1992)
16. Semeniuta, S., Severyn, A., Barth, E.: A hybrid convolutional variational autoencoder for text generation. In: Proceedings of Empirical Methods in Natural Language Processing (2017)
17. Sønderby, C.K., Raiko, T., Maaløe, L., Sønderby, S.K., Winther, O.: Ladder variational autoencoders. Adv. Neural Inf. Process. Syst. **29**, 3738–3746 (2016)
18. Wu, J., Hu, C., Wang, Y., Hu, X., Zhu, J.: A hierarchical recurrent neural network for symbolic melody generation (2017)