A CONTEXTUAL LATENT SPACE MODEL: SUBSEQUENCE MODULATION IN MELODIC SEQUENCE

Taketo Akama
Sony Computer Science Laboratories, Tokyo, Japan
taketo.akama@sony.com

ABSTRACT

Some generative models for sequences such as music and text allow us to edit only subsequences, given surrounding context sequences, which plays an important part in steering generation interactively. However, editing subsequences mainly involves randomly resampling subsequences from a possible generation space. We propose a contextual latent space model (CLSM) in order for users to be able to explore subsequence generation with a sense of direction in the generation space, e.g., interpolation, as well as exploring variations—semantically similar possible subsequences. A context-informed prior and decoder constitute the generative model of CLSM, and a context position-informed encoder is the inference model. In experiments, we use a monophonic symbolic music dataset, demonstrating that our contextual latent space is smoother in interpolation than baselines, and the quality of generated samples is superior to baseline models. The generation examples are available online. [1]

1. INTRODUCTION

Deep generative models permit sequences of decent quality to be generated such as music, lyrics, or text, where standard models generate sequences by sampling from left to right. However, to make creative works in human-machine collaborative settings, controllability—such as modifying unsatisfactory portions with specified intentions—should be improved.

Two major model classes of controllability are (i) latent space models [1–6] and (ii) positional constraint models [7–11]. Latent space models enable us to obtain variations or morphing/interpolations between generated sequences. Positional constraint models, on the other hand, allow us to resample a subsequence without changing the rest of the sequence (*context sequences*), despite the fact that subsequences are sampled randomly and cannot be controlled with morphing/interpolation or variations. Each class of models has its own benefits for making generation systems flexible.

[1] https://contextual-latent-space-model.github.io/demo/

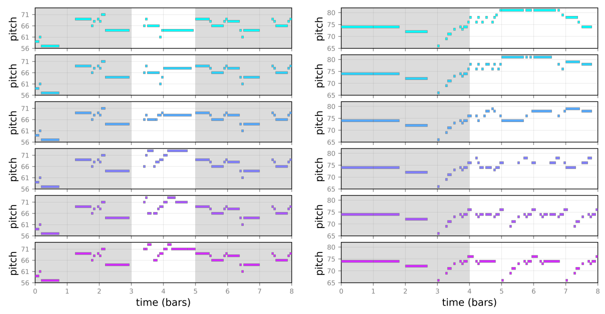
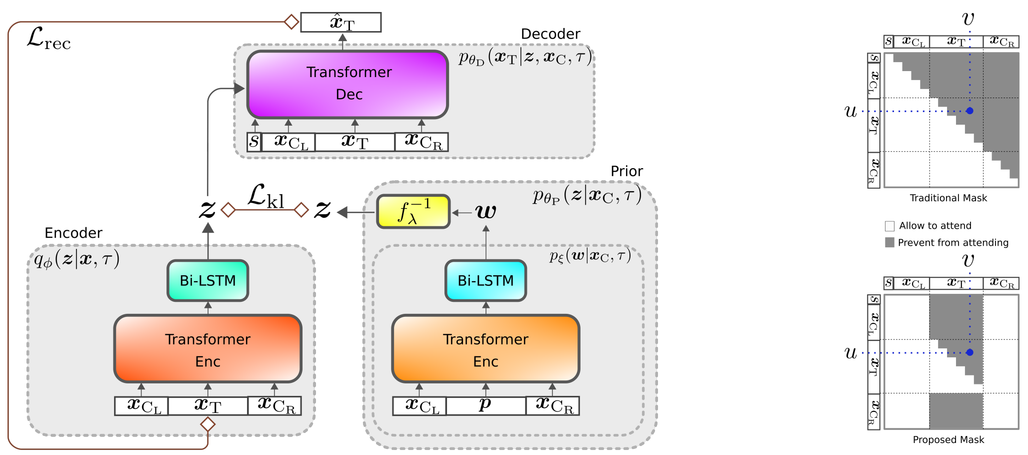


Figure 1: **Contextual Interpolation Examples.** Shaded regions are contexts. For both left and right figures, our CLSM first generates top and bottom melodies under constraints of contexts, and then it generates middle four interpolated points in way that is consistent with contexts. CLSM with $\beta = 0.012$ is used.

Can we build a hybrid model that enjoys the best of both worlds as a step towards multifaceted controllability? We propose a contextual latent space model (CLSM) that allows for positional constraints while at the same time enables latent space exploration such as interpolation or variation. An example usage of CLSM's interpolation is narrowing down the candidates of generated subsequences given context sequences. CLSM variation can be used for obtaining minor modifications of subsequences selected among generated ones, given context sequences.

Our approach is based on the framework of variational inference, where our CLSM is composed of prior and decoder models for the generative model and an encoder model for the inference model. The prior model of CLSM outputs a latent distribution given context sequences. We refer to the support of the distribution as the *contextual latent space*. The decoder model of CLSM generates subsequences that fit in with the context, given corresponding points in the contextual latent space. Finally the encoder model infers the latent space distribution given the entire sequence.

We show the effectiveness of our approach using monophonic sequences in the Lakh MIDI dataset, a large symbolic music dataset [12]. Compared with the baseline methods, our CLSM achieves better performance in terms of the smoothness of the latent space and negative log-



$\mathcal{L}_{\text{rec}}$
$\hat{\boldsymbol{x}}_{\text{T}}$
Decoder
$p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau)$
Transformer Dec
S $\boldsymbol{x}_{\text{C}_{\text{L}}}$ $\boldsymbol{x}_{\text{T}}$ $\boldsymbol{x}_{\text{C}_{\text{R}}}$
Prior
$p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau)$
$\mathcal{L}_{\text{kl}}$
$\boldsymbol{z}$
f_λ^{-1}
$\boldsymbol{w}$
Encoder
$q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau)$
$p_\xi(\boldsymbol{w}|\boldsymbol{x}_{\text{C}}, \tau)$
Bi-LSTM
Transformer Enc
$\boldsymbol{x}_{\text{C}_{\text{L}}}$ $\boldsymbol{x}_{\text{T}}$ $\boldsymbol{x}_{\text{C}_{\text{R}}}$
$\boldsymbol{x}_{\text{C}_{\text{L}}}$ $\boldsymbol{p}$ $\boldsymbol{x}_{\text{C}_{\text{R}}}$
v
u
Traditional Mask
Allow to attend
Prevent from attending
Proposed Mask

(a) Model Overview

(b) Self-Attention Masks for Decoder Model

Figure 2: **Schematic Diagram of Contextual Latent Space Model (CLSM).** (a) Linear or MLP layers are omitted for brevity. (b) Masks indicate whether position u attends to position v.

likelihood of the generated samples. In a listening test, participants favored the generated samples of our CLSM more than those of the baselines.

Our contributions are (I) proposing a problem setting for learning a contextual latent space and providing its solution, (II) proposing novel architectures for the model, e.g., a masking strategy for the decoder and combinations of a transformer and LSTM for the encoder and prior, (III) proposing normalizing flows [13] for conditional priors in order to learn complex conditional priors, (IV) proposing an *interpolation edit distance ratio* to quantitatively assess the smoothness of latent space, and (V) demonstrating reasonable performance with our CLSM in an application for symbolic music generation.

2. METHODOLOGY

2.1 Problem Scenario

Let us consider an i.i.d dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)} = (x_1^{(i)}, ..., x_K^{(i)}) \in \mathcal{A}^K\}_{i=1}^N$ of a sequence of symbols $x_k \in \mathcal{A}$ of length K, where $\mathcal{A}$ is the alphabet set of symbols. We partition each sequence $\boldsymbol{x}$ into three subsequences $\boldsymbol{x}_{\text{C}_{\text{L}}}$, $\boldsymbol{x}_{\text{T}}$, and $\boldsymbol{x}_{\text{C}_{\text{R}}}$, such that $\boldsymbol{x} = \boldsymbol{x}_{\text{C}_{\text{L}}} \oplus \boldsymbol{x}_{\text{T}} \oplus \boldsymbol{x}_{\text{C}_{\text{R}}}$, where $\oplus$ denotes the concatenation of sequences. We refer to subsequences $\boldsymbol{x}_{\text{C}_{\text{L}}}$ and $\boldsymbol{x}_{\text{C}_{\text{R}}}$ as *context sequences* and subsequence $\boldsymbol{x}_{\text{T}}$ as the *target sequence*. Let τ denote a variable representing a set of indexes of the target sequence such that $\tau = \{|\boldsymbol{x}_{\text{C}_{\text{L}}}| + 1, |\boldsymbol{x}_{\text{C}_{\text{L}}}| + 2, ..., |\boldsymbol{x}_{\text{C}_{\text{L}}}| + |\boldsymbol{x}_{\text{T}}|\} \in \mathcal{T}$, where $\mathcal{T}$ is a set of all sets of indexes we would like to model with. For notational simplicity, we introduce the shorthand $\boldsymbol{x}_{\text{C}} = \{\boldsymbol{x}_{\text{C}_{\text{L}}}, \boldsymbol{x}_{\text{C}_{\text{R}}}\}$.

Our goal is to train a generative model with its generative process being (1) $\tilde{\boldsymbol{z}} \sim p(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau)$ and (2) $\tilde{\boldsymbol{x}}_{\text{T}} \sim p(\boldsymbol{x}_{\text{T}}|\tilde{\boldsymbol{z}}, \boldsymbol{x}_{\text{C}}, \tau)$, where $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^{d_z}$ captures the variability of $\boldsymbol{x}_{\text{T}}$ given $\boldsymbol{x}_{\text{C}}$ and τ. We would also like some distance in the latent space of the prior model to represent the similarity of $\boldsymbol{x}_{\text{T}}$ so that the model can be used for e.g., morphing/interpolation or variation generation.

2.2 Model

Fig. 2 is a schematic illustration of our model. Our proposed approach is training a generative model by maximizing the marginal log-likelihood $\log p_\theta(\boldsymbol{x}_{\text{T}}|\boldsymbol{x}_{\text{C}}, \tau) = \log \int p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau) p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau) d\boldsymbol{z}$. Since its computation is intractable in the general case, we introduce the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau)$ to derive the evidence lower bound (ELBO) [14]. Formally,

$$\begin{aligned}
&\log p_\theta(\boldsymbol{x}_{\text{T}}|\boldsymbol{x}_{\text{C}}, \tau) \\
&= \log \int p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau) p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau) d\boldsymbol{z} \\
&= \log \int q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau) \frac{p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau) p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau)}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau)} d\boldsymbol{z} \\
&\geq \int q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau) \log \frac{p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau) p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau)}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau)} d\boldsymbol{z} \\
&= \mathcal{L}_{\text{rec}} - \mathcal{L}_{\text{kl}}, \qquad (1)
\end{aligned}$$

where

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\tau)} \left[\log p_{\theta_{\text{D}}}(\boldsymbol{x}_{\text{T}}|\boldsymbol{z}, \boldsymbol{x}_{\text{C}}, \tau)\right], \qquad (2)$$

$$\mathcal{L}_{\text{kl}} = \text{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau) || p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau)\right). \qquad (3)$$

In practice, we introduce weighting factors in ELBO [15]. Then, our optimization problem is:

$$\max_{\theta,\phi} \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}} \mathbb{E}_{\tau\in\mathcal{T}} \left[\frac{1}{|\boldsymbol{x}_{\text{T}}|} \mathcal{L}_{\text{rec}} - \beta \mathcal{L}_{\text{kl}}\right], \qquad (4)$$

where $\frac{1}{|\boldsymbol{x}_{\text{T}}|}$ is a normalizing factor, and β is a balancing factor of the two terms. The specific choices of β are explained in Sec. 3.4.

Since the conditional prior is generally multimodal and complex, we propose modeling the conditional prior using normalizing flows [13]:

$$p_{\theta_{\text{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\text{C}}, \tau) = p_\xi(\boldsymbol{w}|\boldsymbol{x}_{\text{C}}, \tau) \left|\det\left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{z}}\right)\right|, \qquad (5)$$

$$\boldsymbol{w} = f_\lambda(\boldsymbol{z}) \in \mathcal{W} \subset \mathbb{R}^{d_z}, \qquad (6)$$

where $f_\lambda \colon \mathcal{Z} \to \mathcal{W}$ is an invertible function, and $p_\xi(\cdot|\boldsymbol{x}_{\mathrm{C}}, \tau)$ is a Gaussian distribution. We choose to use affine-coupling layers for f_λ, which was proposed for real-valued non-volume preserving (realNVP) [16].

The Gaussian distribution and the categorical distribution are used for the encoder model $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau)$ and the decoder model $p_{\theta_{\mathrm{D}}}(\boldsymbol{x}_{\mathrm{T}}|\boldsymbol{z}, \boldsymbol{x}_{\mathrm{C}}, \tau)$, respectively.

The network architectures for parametrizing each distribution are explained in Sec. 2.4.

2.3 Applications

2.3.1 *Contextual Interpolation*

Given $\tilde{\boldsymbol{z}}^{(1)}, \tilde{\boldsymbol{z}}^{(2)} \sim p_{\theta_{\mathrm{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\mathrm{C}}, \tau)$, we provide a procedure for generating contextual interpolations between $\tilde{\boldsymbol{x}}^{(1)}$ and $\tilde{\boldsymbol{x}}^{(2)}$, where $\tilde{\boldsymbol{x}}^{(i)} = \boldsymbol{x}_{\mathrm{C_L}} \oplus \tilde{\boldsymbol{x}}_{\mathrm{T}}^{(i)} \oplus \boldsymbol{x}_{\mathrm{C_R}}$ with $\tilde{\boldsymbol{x}}_{\mathrm{T}}^{(i)} \sim p_{\theta_{\mathrm{D}}}(\boldsymbol{x}_{\mathrm{T}}|\tilde{\boldsymbol{z}}^{(i)}, \boldsymbol{x}_{\mathrm{C}}, \tau)$ for $i = 1, 2$.

First, the interpolated latent vector $\tilde{\boldsymbol{z}}(\alpha)$ with a blending ratio of $\alpha \in [0, 1]$ is given by

$$\tilde{\boldsymbol{z}}(\alpha) = f_\lambda^{-1}\left((1-\alpha) f_\lambda(\tilde{\boldsymbol{z}}^{(1)}) + \alpha f_\lambda(\tilde{\boldsymbol{z}}^{(2)})\right), \tag{7}$$

which means that linear interpolation is performed in the $\mathcal{W}$ space, but not in the $\mathcal{Z}$ space, achieving non-linear interpolation in the $\mathcal{Z}$ space. Then, $\tilde{\boldsymbol{z}}(\alpha)$ is decoded and concatenated with the context sequence, yielding $\tilde{\boldsymbol{x}}(\alpha) = \boldsymbol{x}_{\mathrm{C_L}} \oplus \tilde{\boldsymbol{x}}_{\mathrm{T}}(\alpha) \oplus \boldsymbol{x}_{\mathrm{C_R}}$ with

$$\tilde{\boldsymbol{x}}_{\mathrm{T}}(\alpha) \sim p_{\theta_{\mathrm{D}}}(\boldsymbol{x}_{\mathrm{T}}|\tilde{\boldsymbol{z}}(\alpha), \boldsymbol{x}_{\mathrm{C}}, \tau). \tag{8}$$

2.3.2 *Contextual Variation*

Given $\tilde{\boldsymbol{z}} \sim p_{\theta_{\mathrm{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\mathrm{C}}, \tau)$, we provide a procedure for generating contextual variations of $\tilde{\boldsymbol{x}}$, where $\tilde{\boldsymbol{x}} = \boldsymbol{x}_{\mathrm{C_L}} \oplus \tilde{\boldsymbol{x}}_{\mathrm{T}} \oplus \boldsymbol{x}_{\mathrm{C_R}}$ with $\tilde{\boldsymbol{x}}_{\mathrm{T}} \sim p_{\theta_{\mathrm{D}}}(\boldsymbol{x}_{\mathrm{T}}|\tilde{\boldsymbol{z}}, \boldsymbol{x}_{\mathrm{C}}, \tau)$. Letting $\delta \in \mathbb{R}_{\geq 0}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{d_{\mathrm{z}}}$ be a scaling factor of the variation amount and sampled noise from a normal distribution $\mathcal{N}(\boldsymbol{0}, \Sigma)$ with Σ denoting the covariance matrix of $p_\xi(\boldsymbol{w}|\boldsymbol{x}_{\mathrm{C}}, \tau)$, a variation of the latent vector $\tilde{\boldsymbol{z}}$ is given by $\tilde{\boldsymbol{z}}(\delta) = f_\lambda^{-1}\left(f_\lambda(\tilde{\boldsymbol{z}}) + \delta\boldsymbol{\epsilon}\right)$. Then, $\tilde{\boldsymbol{z}}(\delta)$ is decoded and concatenated in the same manner as Sec. 2.3.1.

2.4 Network Architecture

2.4.1 *Encoder Model*

As main architectures, we employ a two-layer "transformer encoder" with relative attention [17, 18], followed by a two-layer bidirectional LSTM network (Bi-LSTM) [19, 20]. $\boldsymbol{x}$ is first embedded and added by a positional embedding before being inputted to the transformer. The transformer uses no masks and a sequence of vectors $\boldsymbol{E} = (\boldsymbol{e}_1, ..., \boldsymbol{e}_K)$ is outputted. Let $\boldsymbol{E}_{\mathrm{T}}$ be a subsequence of $\boldsymbol{E}$, consisting of $\boldsymbol{E}$'s elements, whose indexes are in τ, i.e., $\boldsymbol{E}_{\mathrm{T}} = (\boldsymbol{e}_{|\boldsymbol{x}_{\mathrm{C_L}}|+1}, ..., \boldsymbol{e}_{|\boldsymbol{x}_{\mathrm{C_L}}|+|\boldsymbol{x}_{\mathrm{T}}|})$. Only the subsequence $\boldsymbol{E}_{\mathrm{T}}$ is fed to the Bi-LSTM. Let $\boldsymbol{h}_{\mathrm{l}}$ and $\boldsymbol{h}_{\mathrm{r}}$ denote the last outputs of the Bi-LSTM. They are concatenated to be fed to two multi layer perceptrons (MLPs; each for the mean and covariance of the normal distribution), yielding the encoder model $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \tau) = \mathcal{N}\left(\boldsymbol{z}|\mathrm{MLP}(\boldsymbol{h}_{\mathrm{l}} \oplus \boldsymbol{h}_{\mathrm{r}}), \mathrm{diag}(\frac{1}{2}\exp(\mathrm{MLP}(\boldsymbol{h}_{\mathrm{l}} \oplus \boldsymbol{h}_{\mathrm{r}})))\right)$. The MLPs consist of two layers with a SELU activation in between [21].

Concerning the hyper-parameters for the "transformer encoder," the token embedding size, the hidden size, the number of heads, and the dropout rate are set to 128, 256, 8, and 0.1, respectively. The hidden size and the dropout rate for the Bi-LSTM are set to 256 and 0.1, respectively. The hidden size of the MLP and the number of dimensions of $\boldsymbol{z}$ are set to 512 and 128, respectively.

2.4.2 *Prior Model*

As can be seen in Eq. 5 and Eq. 6, $p_\xi(\boldsymbol{w}|\boldsymbol{x}_{\mathrm{C}}, \tau)$ and $f_\lambda(\boldsymbol{z})$ need to be defined for the prior model.

For $p_\xi(\boldsymbol{w}|\boldsymbol{x}_{\mathrm{C}}, \tau)$, as with the encoder model, a two-layer "transformer encoder" is followed by a Bi-LSTM. Unlike the encoder model, we replace each of the elements in the target sequence $\boldsymbol{x}_{\mathrm{T}}$ with a *positional constraint symbol* p. In other words, $\boldsymbol{x}_{\mathrm{C_L}} \oplus \boldsymbol{p} \oplus \boldsymbol{x}_{\mathrm{C_R}}$ is fed to the "transformer encoder", where $\boldsymbol{p} = (p, p, ..., p)$ is a sequence of positional constraint symbols and $|\boldsymbol{p}| = |\boldsymbol{x}_{\mathrm{T}}|$. The hyper-parameters for the "transformer encoder," the Bi-LSTM, and the number of dimensions of $\boldsymbol{z}$ are set to the same values as in Sec. 2.4.1.

To parameterize $f_\lambda(\boldsymbol{z})$, four-layer affine-coupling layers are employed. Each of the scale and bias networks of the affine-coupling layers consists of a three-layer MLP, where each hidden size is 256, and the activations are leaky ReLUs with a negative slope of 0.01. For each of the scale networks, a $\tanh$ activation is used after the last linear layer.

2.4.3 *Decoder Model*

As a main architecture, we employ a two-layer "transformer decoder" with relative attention. Let s be a symbol representing the start of a sequence. The concatenation $s \oplus \boldsymbol{x}$ is embedded and added by positional embeddings before being inputted to the transformer.

We propose using an effective encoder-decoder attention mechanism for the latent space. Each latent vector $\tilde{\boldsymbol{z}}$ sampled from the encoder model is first fed to a linear layer to map $\tilde{\boldsymbol{z}} \in \mathbb{R}^{d_{\mathrm{z}}}$ to $\tilde{\boldsymbol{z}}' \in \mathbb{R}^{d_{\mathrm{z}} l_{\mathrm{z}}}$, which is reshaped to form $\tilde{\boldsymbol{Z}} \in \mathbb{R}^{d_{\mathrm{z}} \times l_{\mathrm{z}}}$, a sequence of l_{z} vectors, where the dimensionality of each vector is d_{z}. The sequence $\tilde{\boldsymbol{Z}}$ is then attended to by the transformer by means of encoder-decoder attention. We set $l_{\mathrm{z}} = 4$ in experiments.

We propose a masking strategy for modeling the decoder, as illustrated in the bottom of Fig. 2b. The positions whose inputs are s, $\boldsymbol{x}_{\mathrm{C_L}}$, and $\boldsymbol{x}_{\mathrm{C_R}}$ are allowed to attend to positions except those of $\boldsymbol{x}_{\mathrm{T}}$. On the other hand, positions whose inputs are $\boldsymbol{x}_{\mathrm{T}}$ are allowed to attend to positions except the future positions of $\boldsymbol{x}_{\mathrm{T}}$.

Let the output of the transformer denote $\boldsymbol{D} = (\boldsymbol{d}_1, ..., \boldsymbol{d}_K)$. Let $\boldsymbol{D}_{\mathrm{T}}$ be a subsequence of $\boldsymbol{D}$, consisting of $\boldsymbol{D}$'s elements whose indexes are in τ, i.e., $\boldsymbol{D}_{\mathrm{T}} = (\boldsymbol{d}_{|\boldsymbol{x}_{\mathrm{C_L}}|+1}, ..., \boldsymbol{d}_{|\boldsymbol{x}_{\mathrm{C_L}}|+|\boldsymbol{x}_{\mathrm{T}}|})$. Let $\boldsymbol{x}_{\mathrm{T}}[i]$ and $\boldsymbol{D}_{\mathrm{T}}[i]$ denote the ith elements of $\boldsymbol{x}_{\mathrm{T}}$ and $\boldsymbol{D}_{\mathrm{T}}$, respectively. Then the decoder model is defined as follows: $p_{\theta_{\mathrm{D}}}(\boldsymbol{x}_{\mathrm{T}}|\boldsymbol{z}, \boldsymbol{x}_{\mathrm{C}}, \tau) = \prod_{i=1}^{|\boldsymbol{x}_{\mathrm{T}}|} \mathrm{Cat}(\boldsymbol{x}_{\mathrm{T}}[\mathrm{i}]|\mathrm{Softmax}(\mathrm{Linear}(\boldsymbol{D}_{\mathrm{T}}[\mathrm{i}])))$, where Cat and Linear denote a categorical distribution and a linear layer respectively. The hyper-parameters of the "transformer decoder" are set to be equal to those of the "transformer encoder" in Sec. 2.4.1.

3. EXPERIMENTAL SETUP

3.1 Dataset

We created datasets from LMD-matched of the Lakh MIDI dataset [12], comprising 45,129 files matched to the song identity entries in the Million Song Dataset [22]. Each song has one or several different versions of MIDI files. We first extracted files with a 4/4 time signature, used the accompanying tempo information to determine beat locations, and quantized each beat into 4. We then split the song identities into a proportion of 11:1:6:1:1 to create train-1, validation-1, train-2, validation-2, and test datasets, respectively. The train-1 and validation-1 datasets were for training the proposed and baseline models, whereas the train-2 and validation-2 datasets were for training evaluation models. The test dataset was for input sequences when evaluating models. We filtered out non-monophonic tracks, bass or drum tracks, and tracks outside the pitch range of [55, 84]. We conducted data augmentation by transposing tracks to all possible keys if the transposed tracks stayed within the pitch range of [55, 84]. We retrieved 8-bar sliding windows (with a stride of 1 bar) from each track followed by filtering out windows that had more than one bar of consecutive rests. For encoding musical sequences, we adopted the melodico-rhythmic encoding proposed in [9], where we used the pitch of musical notes as symbols "55",...,"84" and used "R" to represent a rest symbol. We added an extra symbol, "__" representing that a note is held and not replayed.

3.2 Baseline Methods

3.2.1 VAE

We trained a VAE [14], in which a two-layered Bi-LSTM and LSTM were used for the encoder and decoder, respectively. The decoder, encoder, and prior models used the categorical, normal, and standard normal distribution, respectively. The optimization problem is

$$\max_{\psi,\omega} \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}} \left[\frac{1}{|\boldsymbol{x}|} \left(\mathcal{L}_{\mathrm{rec}}^{\mathrm{VAE}} - \gamma \mathcal{L}_{\mathrm{kl}}^{\mathrm{VAE}} \right) \right], \qquad (9)$$

where $\mathcal{L}_{\mathrm{rec}}^{\mathrm{VAE}} = \mathbb{E}_{q_\omega(\boldsymbol{z}|\boldsymbol{x})}[\log p_\psi(\boldsymbol{x}|\boldsymbol{z})]$, $\mathcal{L}_{\mathrm{kl}}^{\mathrm{VAE}} = \mathrm{KL}(q_\omega(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$, and $\frac{1}{|\boldsymbol{x}|}$ is a normalizing factor.

Given $\boldsymbol{x}_{\mathrm{C}}$, τ, and $\tilde{\boldsymbol{z}}^{(1)}, \tilde{\boldsymbol{z}}^{(2)} \sim p(\boldsymbol{z})$, interpolation is conducted as follows. First, interpolated latent vector $\tilde{\boldsymbol{z}}(\alpha)$ is given by

$$\tilde{\boldsymbol{z}}(\alpha) = (1-\alpha)\tilde{\boldsymbol{z}}^{(1)} + \alpha\tilde{\boldsymbol{z}}^{(2)}. \qquad (10)$$

Then, $\tilde{\boldsymbol{z}}(\alpha)$ is decoded and concatenated with the context sequence: $\tilde{\boldsymbol{x}}(\alpha) = \boldsymbol{x}_{\mathrm{C_L}} \oplus \tilde{\boldsymbol{x}}_{\mathrm{T}}(\alpha) \oplus \boldsymbol{x}_{\mathrm{C_R}}$ with

$$\tilde{\boldsymbol{x}}_{\mathrm{T}}(\alpha) \sim p_\psi(\boldsymbol{x}_{\mathrm{T}}|\tilde{\boldsymbol{z}}(\alpha), \boldsymbol{x}_{\mathrm{C_L}}), \qquad (11)$$

where $p_\psi(\boldsymbol{x}_{\mathrm{T}}|\boldsymbol{z}, \boldsymbol{x}_{\mathrm{C_L}})$ can be immediately obtained from the autoregressive decoder model of VAE. Note that the proposed CLSM has advantages over VAE in terms of probabilistic dependencies (i) the decoder model of VAE does not have dependencies on the right context $\boldsymbol{x}_{\mathrm{C_R}}$, and (ii) the prior model of VAE does not have dependencies on either the left context $\boldsymbol{x}_{\mathrm{C_L}}$ or the right context $\boldsymbol{x}_{\mathrm{C_R}}$. The property of (i) motivates us to separately quantify the performance of models in two cases: (1) the case where only the left context exists, and (2) otherwise. In Sec. 4.1, Sec. 4.2, and Fig. 3b, we report the performance of these two cases separately.

Random generation was conducted as follows. First, a latent vector was sampled from the prior distribution. Then, $\tilde{\boldsymbol{z}}$ was decoded and concatenated with context sequences in the same manner as the interpolation.

The hyper-parameters of the LSTMs in VAE were set to be equal to those of the LSTMs in CLSM.

3.2.2 ARNN

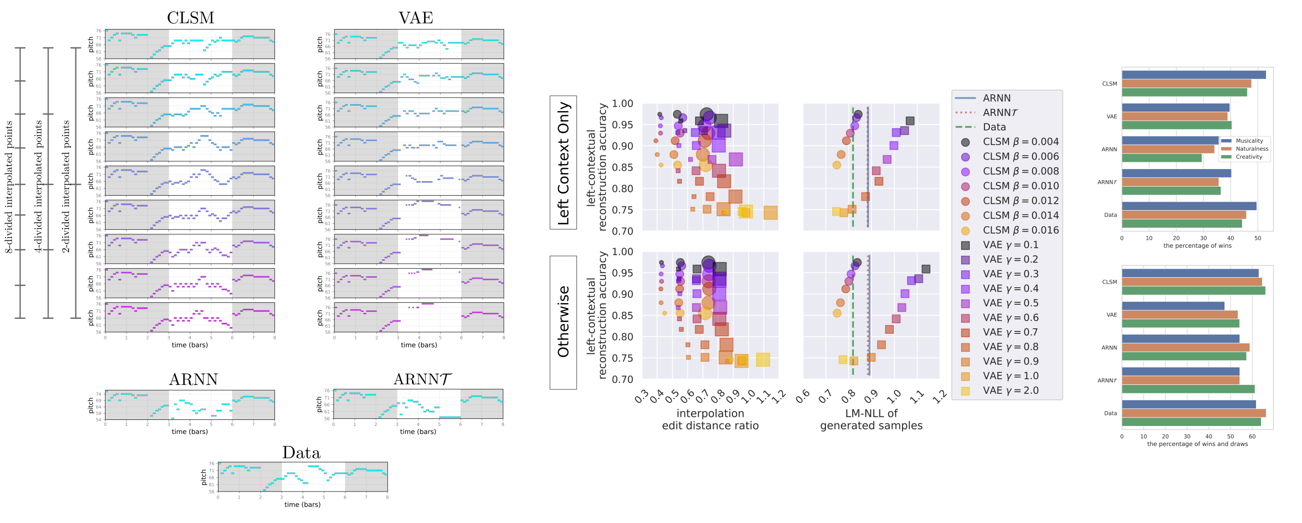
Anticipation RNN (ARNN) [9] is a sequence generation model with positional constraints. Two-layered LSTMs were used for both Token-RNN and Constraint-RNN. The hyper-parameters of the LSTMs were set to be equal to those of the LSTMs in CLSM.

3.3 2D Plane for Comparing CLSM and VAE

The balancing factors β and γ of CLSM and VAE consist in adjusting the trade-off between the reconstruction accuracy and other model performances. Which balancing factors of CLSM correspond to which ones of VAE? It is natural to compare models of similar reconstruction accuracies or compare models of similar performances (except reconstruction accuracies). Therefore, in Sec. 4.1, Sec. 4.2, and Fig. 3b, we plot reconstruction accuracies versus other performance metrics in 2D planes, where the more the plotted point is in the upper left corner, the better it is. As discussed in Sec. 3.2.1, CLSM and VAE have dependencies at least on the left context. To make the reconstruction accuracies of the CLSM and VAE mean basically identical, we used the *left-contextual reconstruction accuracy*, which is the average of the reconstruction accuracies of sequences where target sequences end with index $|\boldsymbol{x}|$, i.e., there is no right context and only the left context exists.

3.4 Training Settings

For all models, teacher forcing was used, the batch size was set to 64, and training was conducted for 2 epochs, when the losses converged. The Adam optimizer [23] was used for all models, with the parameters $(\alpha, \beta_1, \beta_2) = (0.0005, 0.9, 0.999)$. For CLSM or VAE, we conducted



(a) Generation Comparison (b) Objective Evaluation (c) Human Evaluation

Figure 3: **Generation Comparison and Experimental Results.** (a) Shaded regions are contexts. For CLSM and VAE, $\beta = 0.012$ and $\gamma = 0.4$ are used. (b) For evaluating smoothness (to left), small, medium, and large marker are plots for number of divisions in interpolation $J = 8, 4, 2$, respectively. See Sec. 4.1 and 4.2. (c) See Sec. 4.3.

KL-annealing linearly from $\beta = 0$ or $\gamma = 0$ for 2 epochs [1].

3.4.1 *CLSM (Ours)*

At every iteration of the training, indexes of the target sequence $\tau = \{|\boldsymbol{x}_{\mathrm{C_L}}|+1, |\boldsymbol{x}_{\mathrm{C_L}}|+2, ..., |\boldsymbol{x}_{\mathrm{C_L}}|+|\boldsymbol{x}_{\mathrm{T}}|\} \in \mathcal{T}$ should be sampled. In the experiments, we chose to use either 1, 2, 3, or 4 bars as the target sequence length and used a stride of 1 bar for the starting index of the target sequence. Precisely, we sampled (i) $|\boldsymbol{x}_{\mathrm{T}}|$ uniformly from $\{16, 32, 48, 64\}$ and then (ii) sampled $|\boldsymbol{x}_{\mathrm{C_L}}|$ uniformly from $\{0, 16, 32, ..., 128 - |\boldsymbol{x}_{\mathrm{T}}|\}$. Note that 128 corresponds to 8 bars (the sequence length) and that 16 corresponds to 1 bar. The balancing factor β in Eq. 4 is set to $\{0.004, 0.006, ..., 0.016\}$ in Secs. 4.1, 4.2 and 0.012 in Sec. 4.3. The expectations of the two terms in Eq. 4 were approximated with one sample from the encoder model.

3.4.2 *VAE (Baselines)*

The balancing factor γ in Eq. 9 is set to $\{0.1, 0.2, ..., 1.0, 2.0\}$ in Secs. 4.1, 4.2 and 0.4 in Sec. 4.3. The expectations of the reconstruction loss were approximated with one sample from the encoder model. The KL loss term was computed analytically.

3.4.3 *ARNN (Baseline)*

The vanilla ARNN is capable of imposing constraints of any positions. Since it would be possible that restricting constraints to those of our $\mathcal{T}$ would be advantageous when evaluated over $\mathcal{T}$, we also trained and evaluated this model, which we refer to as ARNN$\mathcal{T}$.

3.5 Generation Settings

For CLSM and VAE, each element of sequences was sampled by applying the argmax operation to the categorical distributions of the decoders. For ARNN, multinomial sampling with a temperature of 1.0 was used.

4. EXPERIMENTS AND RESULTS

4.1 Smoothness Analysis in Latent Space

To assess the smoothness of our latent space, we propose the *interpolation edit distance ratio* $R(J)$, which is the ratio of the distance between adjacent interpolated points (sequences) to the distance of interpolation end points (sequences). Formally, $R(J)$ is the normalized average edit distance $d_{\mathrm{edit}}(\cdot,\cdot)$ of adjacent points in J-divided interpolated points:

$$R(J) = \mathbb{E}\left[\frac{\sum_{j=0}^{J-1} d_{\mathrm{edit}}\left(\tilde{\boldsymbol{x}}_{\mathrm{T}}\left(\frac{j}{J}\right), \tilde{\boldsymbol{x}}_{\mathrm{T}}\left(\frac{j+1}{J}\right)\right)}{D(J-n)}\right], \quad (12)$$

where $\tilde{\boldsymbol{x}}_{\mathrm{T}}(\cdot)$ is defined by Eqs. 7, 8 for CLSM and Eqs. 10, 11 for VAE, while D is the edit distance between end points defined by $D = d_{\mathrm{edit}}\left(\tilde{\boldsymbol{x}}_{\mathrm{T}}(0), \tilde{\boldsymbol{x}}_{\mathrm{T}}(J)\right)$. Here, the expectation is approximated by sampling $\tilde{\boldsymbol{z}}^{(1)}, \tilde{\boldsymbol{z}}^{(2)} \sim p_{\theta_{\mathrm{P}}}(\boldsymbol{z}|\boldsymbol{x}_{\mathrm{C}}, \tau)$ for CLSM and $\tilde{\boldsymbol{z}}^{(1)}, \tilde{\boldsymbol{z}}^{(2)} \sim p(\boldsymbol{z})$ for VAE, where 1K samples of $\boldsymbol{x}$ are uniformly sampled from the test dataset, and, for each $\boldsymbol{x}$, τ is uniformly sampled from $\mathcal{T}$. Since the edit distance $d_{\mathrm{edit}}\left(\tilde{\boldsymbol{x}}_{\mathrm{T}}\left(\frac{j}{J}\right), \tilde{\boldsymbol{x}}_{\mathrm{T}}\left(\frac{j+1}{J}\right)\right)$ sometimes becomes zero, we excluded cases through division by $J-n$ instead of J as in Eq. 12, where n is the number of edit distances that are zero among $j = 0, 1, ..., J-1$. We also excluded cases where $D(J-n) = 0$. The left scatter plot of Fig. 3b shows a comparison of CLSM and VAE. The small, medium, and large markers are plots for the number of divisions in interpolation $J = 8, 4, 2$, respectively. For each J, CLSM performed better than VAE. The plots of VAE $J = 8$ overlap with those of CLSM $J = 4$, indicating that even 4-divided interpolation of CLSM was as smooth as 8-divided interpolation of VAE. The results are similar in the two cases of “left context only” and “otherwise.”

4.2 NLL Evaluation of Generated Samples

To evaluate the quality of generated samples, we computed the negative log-likelihood (NLL) of the generated samples. NLL was computed by using a separately trained vanilla transformer language model (LM) that had two layers and was autoregressive. The LM was trained by using the train-2 and validation-2 datasets defined in Sec. 3.1. The samples to be evaluated were the same as the 8-divided interpolated points in Sec. 4.1. Since ARNN is not capable of interpolation, only a random sampling of two points was performed per each context for ARNN. The right scatter plot of Fig. 3b shows a comparison of the CLSM, VAE, ARNNs, and the test dataset (Data). CLSM outperformed VAE by a large margin when we compared models that were close in terms of reconstruction accuracy or NLL—a reasonable strategy of comparison as we discuss in Sec. 3.3. Compared with the ARNNs, the performance of CLSM was superior in all settings. The results of VAE were better in the case of “left context only” than in the case of “otherwise.” In contrast, for the CLSM and ARNNs, there were only slight differences between the two cases. This empirically demonstrates that the decoder model of VAE has dependencies only on the left context but not on the right context, as we discuss in Sec. 3.2.1.

4.3 Human Evaluation

To further assess the quality of sequences generated by each model, we conducted listening tests using Amazon Mechanical Turk. We sampled 32 sequences from the test dataset, for which context positions were randomly sampled, and the target sequence lengths were sampled from 1-4 bars. We conducted a pair-wise comparison of the generation results of each model using the same context sequences. We considered all possible combinations, yielding 320 pair-wise comparisons. The order within each pair was randomized. We chose to use $\beta = 0.012$ for CLSM, since the performance trade-offs are well balanced according to the results in Sec. 4.1 and Sec. 4.2. As discussed in Sec. 3.3 since it is reasonable to choose a VAE model with a similar reconstruction accuracy to that of CLSM, we chose $\gamma = 0.4$ for VAE. Participants with different levels of musical expertise were asked to rate “which music is better in terms of musicality, naturalness, and creativity” on a Likert scale. Fig. 3c shows a comparison of the CLSM, VAE, ARNNs, and the test dataset (Data). CLSM and Data performed the best in terms of the percentage of wins (Fig. 3c, top) as well as the percentage of wins and draws (Fig. 3c, bottom). Interestingly, the ARNNs tended to outperform VAE when the number of draws was included, whereas VAE tended to outperform the ARNNs when only the number of wins was considered. This might indicate that the performance of VAE tends to be extreme.

5. RELATED WORK

T-CVAE is a transformer-based conditional VAE model for story completion [24]. VAEAC [25] is a CNN- or MLP-based VAE that enables us to impose any positional constraints. Although their probabilistic frameworks are similar to ours, the models and architectures are quite different. Unlike their models, CLSM is demonstrated to perform interpolation in the latent space. Moreover, the data domain of T-CVAE is story text, and the domains of VAEAC are image and feature classification/regression datasets, while ours is for sequence datasets and experimented on music.

Although contexts are not considered for latent variables, there are several works that use transformers for learning a global latent variable for sequences using AE or VAE. For text-style transfer, a “transformer encoder” outputs are all fed to GRU to yield a latent vector, which is attended to by a decoder [26]. To learn the styles of piano performances, a “transformer encoder” outputs are summed to be attended to by a decoder [27]. In OPTIMUS for sentence modulation [28] and INSET [29] for sentence infilling, a CLS token is additionally fed to a “transformer encoder,” and the output at the position of the CLS yields a latent vector, which is fed to a decoder either by self-attention and/or by being added to word embeddings of the decoder (OPTIMUS) or by being inputted as the first token (INSET).

For language processing, the authors of UniLM propose seq-to-seq LM, where they divide a whole sequence into first and second segments [30]. The self-attention masks are bidirectional and unidirectional for the first and second segments, respectively. Our decoder mask is different in that it divides a sequence into three segments, where the length of the first and third segments can be zero during the training or inference phase. Also, the training procedure of them is BERT-like, which is different from ours [31].

RealNVP has been used for the prior in VAE in order to improve the performance of VAE [32, 33]. However, these works are not only in the domain of images but also use non-conditional priors, which differs from ours.

6. CONCLUSION

We proposed a contextual latent space model (CLSM), in which the left and/or right contexts of sequences can be constrained to generate interpolations or variations. A context-informed prior and decoder constitute the generative model of CLSM and a context position-informed encoder is the inference model.

The latent space of CLSM was quantitatively shown to be smoother than baselines. Furthermore, the generation fidelity was demonstrated to be superior to the baseline methods. It would be useful to apply our approach to other data domains such as polyphonic music, lyrics, or text. The benefits of the latent space model are not only enabling interpolations and variations but also enabling transformations of attributes or style transfer. It would be desirable to extend our approach to these kinds of applications.

7. REFERENCES

[1] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

[2] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the 35th International Conference on Machine Learning*, 2018.

[3] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019.

[4] A. Pati and A. Lerch, "Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders," *Neural Computing and Applications*, 2020.

[5] H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *ISMIR*, 2020.

[6] T. Akama, "Connective fusion: Learning transformational joining of sequences with application to melody creation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020.

[7] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[8] A. Wang and K. Cho, "BERT has a mouth, and it must speak: BERT as a markov random field language model," *CoRR*, vol. abs/1902.04094, 2019.

[9] G. Hadjeres and F. Nielsen, "Anticipation-rnn: enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, vol. 32, no. 4, pp. 995–1005, 2020.

[10] C. A. Huang, T. Cooijmans, A. Roberts, A. C. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.

[11] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," *ISMIR*, 2019.

[12] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

[13] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.

[15] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR*, 2017.

[16] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *5th International Conference on Learning Representations, ICLR 2017*.

[17] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

[18] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *International Conference on Learning Representations*, 2019.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[20] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[21] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017.

[22] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[24] T. Wang and X. Wan, "T-cvae: Transformer-based conditioned variational autoencoder for story completion," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019.

[25] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning," in *International Conference on Learning Representations*, 2019.

[26] K. Wang, H. Hua, and X. Wan, "Controllable unsupervised text attribute transfer via editing entangled latent representation," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[27] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[28] C. Li, X. Gao, Y. Li, X. Li, B. Peng, Y. Zhang, and J. Gao, "Optimus: Organizing sentences via pre-trained modeling of a latent space," in *EMNLP*, 2020.

[29] Y. Huang, Y. Zhang, O. Elachqar, and Y. Cheng, "INSET: sentence infilling with inter-sentential transformer," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., 2020.

[30] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[31] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019.*

[32] C.-W. Huang, A. Touati, L. Dinh, M. Drozdzal, M. Havaei, L. Charlin, and A. Courville, "Learnable explicit density for continuous latent space and variational inference," 2017.

[33] H. Xu, W. Chen, J. Lai, Z. Li, Y. Zhao, and D. Pei, "On the necessity and effectiveness of learning the prior of variational auto-encoder," 2019.