CONNECTIVE FUSION: LEARNING TRANSFORMATIONAL JOINING OF SEQUENCES WITH APPLICATION TO MELODY CREATION

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

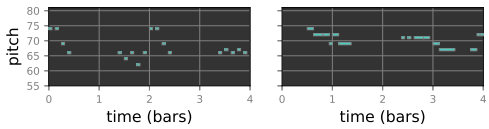
ABSTRACT

We present *Connective Fusion*, a music generation scheme by transformational joining of two musical sequences for creative purposes. Given two shorter sequences as inputs, our model transforms each of them such that their concatenation is more coherent to form a longer sequence, while each of the transformed shorter sequences retains meaningful similarity with the corresponding input sequence. In short, our model connects and fuses two contextually unrelated sequences in a coherent way. This transformation can be applied iteratively to gradually fuse the input sequences. The *style latent space* is simultaneously learned, allowing users to control how the two sequences are merged. Our approach comprises two steps of unsupervised learning: a deep generative model with a latent space is learned, followed by adversarial learning of the transformation function in the latent space. We demonstrate the usefulness of our method through the task of melody creation using a symbolic music dataset.

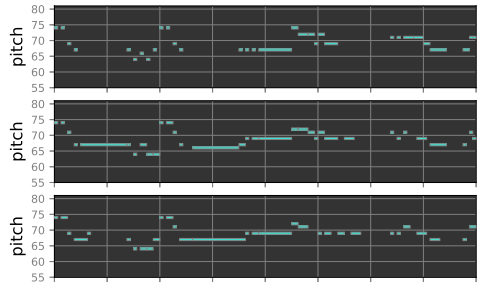
1. INTRODUCTION

Spurred by the progress of deep neural networks, symbolic music generation systems, especially the ones with user controllability have gathered renewed interests these days. Most of the systems with controllability can be categorized into generating continuation [1], regenerating arbitrary positions [2, 3], unsupervised conditional generation [4, 7], or transforming musical sequences such that musical attributes, concepts, or styles are altered [4–8].

In this paper, we seek another category, generation by fusing input musical sequences as ideas. Specifically, we propose *Connective Fusion*, a generative transformation scheme which allows two input musical sequences to be transformed such that the concatenated musical sequence is musically plausible (See Fig.1 for the illustration). After the transformation, each of the two sequences has meaningful similarity with the one before the transformation. Two input musical sequences can also be transformed iteratively to gradually increase the coherency of joining (Fig.1). Our generative transformation differs from pure transformation in that it permits users to explore or sample from the *style space* of how the two sequences are combined (Fig.2). The application of Connective Fusion includes i) providing users with musical ideas based on not

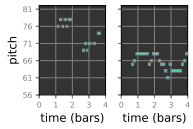


(a) Given inputs: random combination of two sequences

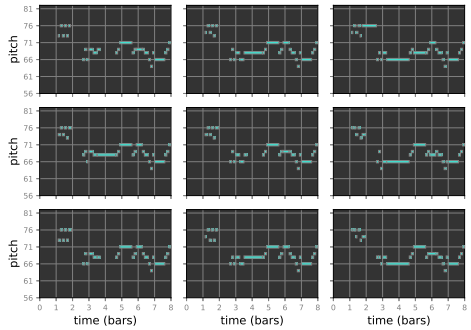


(b) Transformed outputs of iteration 1,2, and 3

Figure 1: Iterated application of transformation.



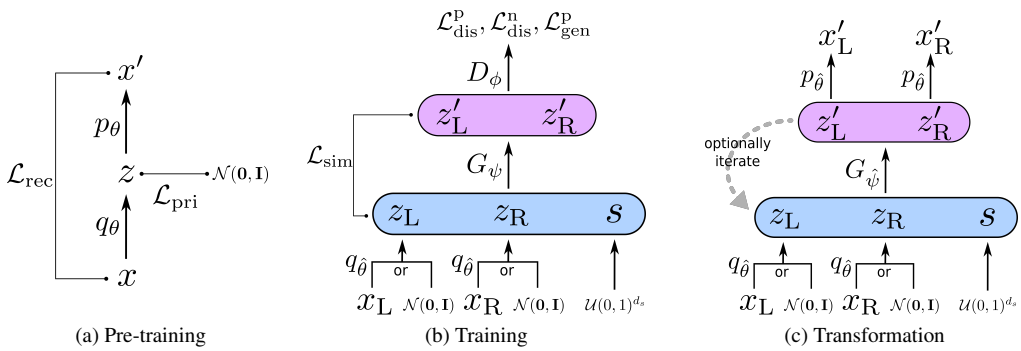
(a) Given inputs: random combination of two sequences



(b) Transformed outputs

Figure 2: Transformation with *style space* exploration.

© T. Akama. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** T. Akama, "Connective Fusion: Learning Transformational Joining of Sequences with Application to Melody Creation", in *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, Montréal, Canada, 2020.



(a) Pre-training (b) Training (c) Transformation

Figure 3: Connective Fusion schematic diagrams.

necessarily polished ideas on hand, and ii) creating novel musical flows by combining and fusing musical fragments of different characteristics in a coherent way.

Technically, our approach is training a transformation function in an unsupervised learning manner, consisting of two steps: the pre-training step and the training step. In the pre-training step, a Variational Auto-Encoder (VAE) [9] is trained to obtain mappings between the *data space* and the *latent space*. A given pair of musical sequences are fed into the encoder of the VAE to be mapped to the corresponding latent vectors in the latent space. In the following training step, models are trained upon the representation of the VAE latent space. The transformation function, the *generator* is trained adversarially [10] such that transformed results are indistinguishable from the human-made musical sequences. Together with the adversarial loss, the similarity loss, consisting of the latent space distance before and after the transformation, is simultaneously taken into account in order to train models with desired amount of transformation.

In experiments, we empirically demonstrate that our method outperforms a baseline method with various parameter settings in terms of reality and five musical statistics at each transformation amount. We also quantitatively show that iterated application of our transformation allows gradual transformation while having comparable to or better performance than single (non-iterative) transformation.

Our contributions of this paper are: i) proposing a new problem setting/task and its solution, ii) presenting a model and procedure for learning style space, iii) introducing iterated application of transformation for gradual transformation, and iv) demonstrating the performance and the application for melody creation.

2. METHODOLOGY

2.1 Problem Scenario

Suppose we have a dataset of sequences $\mathcal{D}_y = \{y^{(i)} = x_{\mathrm{L}}^{(i)} \oplus x_{\mathrm{R}}^{(i)}\}_{i=1}^{N}$, where $y^{(i)} \in \mathcal{Y}$ is the concatenation of two subsequences $x_{\mathrm{L}}^{(i)} \in \mathcal{X}$ and $x_{\mathrm{R}}^{(i)} \in \mathcal{X}$. We can consider $\mathcal{D}_y$ as a sequence pair dataset $\mathcal{D} = \{(x_{\mathrm{L}}^{(i)}, x_{\mathrm{R}}^{(i)})\}_{i=1}^{N}$. For instance, $y^{(i)}$ is a musical sequence of 8 bars, whereas $x_{\mathrm{L}}^{(i)}$ and $x_{\mathrm{R}}^{(i)}$ are musical sequences of 4 bars.

Using the dataset $\mathcal{D}$, the scheme of Connective Fusion basically solves the following task: given two sequences $x_{\mathrm{L}}, x_{\mathrm{R}} \in \mathcal{X}$, transforming x_{L} to x'_{L} and x_{R} to x'_{R} such that the concatenated sequence $x'_{\mathrm{L}} \oplus x'_{\mathrm{R}}$ becomes more probable to be a sample in $\mathcal{Y}$ than $x_{\mathrm{L}} \oplus x_{\mathrm{R}}$, while the transformed sequences $x'_{\mathrm{L}}, x'_{\mathrm{R}}$ retain meaningful similarity to the given sequences $x_{\mathrm{L}}, x_{\mathrm{R}}$, respectively. For example, this serves as a solution for generating longer musical sequences given shorter musical sequence pairs of any combinations as inspiration, which is useful for composing new music based on unpolished musical ideas on hand.

2.2 Approach

Schematic diagrams of pre-training, training, and transformation of Connective Fusion are depicted in Fig.3.

2.2.1 Pre-training

Our approach is first training a Variational Auto-Encoder (VAE) model for obtaining bidirectional mappings between each sequence $x \in \mathcal{X}$ and its latent vector $z \in \mathcal{Z} \subset \mathbb{R}^{d_z}$, which has the compressed information of x.

2.2.2 Training

Then we train a generator G that transforms any pair $(z_{\mathrm{L}}, z_{\mathrm{R}}) \in \mathcal{Z} \times \mathcal{Z}$ to $(z'_{\mathrm{L}}, z'_{\mathrm{R}}) \in \mathcal{Z} \times \mathcal{Z}$ that are indistinguishable from the pairs in the dataset. The generator also takes a style vector $s \in \mathcal{S}$ from the style space $\mathcal{S} = [0,1]^{d_s}$ as an additional input. Formally, the generator can be written as $(z'_{\mathrm{L}}, z'_{\mathrm{R}}) = G(z_{\mathrm{L}}, z_{\mathrm{R}}, s)$ and $G\colon \mathcal{Z} \times \mathcal{Z} \times \mathcal{S} \to \mathcal{Z} \times \mathcal{Z}$. The generator G is trained together with the discriminator $D\colon \mathcal{Z} \times \mathcal{Z} \to [0,1]$ with the adversarial learning framework [10]. In addition to the adversarial loss, we add the similarity loss in order to adjust the degree of similarity before and after the transformation.

2.2.3 Transformation

Given two sequences x_{L} and x_{R}, these sequences are first fed to the encoder of the VAE to obtain z_{L} and z_{R}, respectively. The latent vectors z_{L} and z_{R} together with a style vector $s \in \mathcal{S}$ sampled from $\mathcal{S} = [0,1]^{d_s}$ are then fed to the

generator G to transform z_{L}, z_{R} to z'_{L}, z'_{R}, respectively. Finally, the transformed latent vectors z'_{L}, z'_{R} are fed to the decoder to generate x'_{L}, x'_{R}, respectively. The concatenation $x'_{\mathrm{L}} \oplus x'_{\mathrm{R}}$ is the generated sequence. Instead, z_{L} and z_{R} can also be sampled from the prior distribution $p(z)$, providing users with scratch generation functionality followed by the transformation functionality.

The style space $\mathcal{S}$ allows us to control *how the two sequences are connectively fused*. For example, as illustrated in Fig.2 and explained in the experiments Sec.3.3, users can choose preferred styles of fusion among interpolated styles on a 2D plane.

We also propose iterated transformation with G, which consists in applying G repeatedly, drawing two interrelated trajectories in the latent space. The higher the number of iterations becomes, the farther the latent vectors tend to move from the original position. This is useful for users to control the degree of similarity in transformation.

2.3 Advantage of Approach Using Latent Space

Using the representation in $\mathcal{Z}$ but not the one in $\mathcal{X}$ i) is useful for defining similarity before and after the transformation, ii) is computationally inexpensive, and iii) bypasses the difficult problem of back-propagating through discrete sampling of sequences.

2.4 Pre-training Detail: Encoder and Decoder Training with Auto-Encoding VB algorithm

To train the VAE, another dataset $\mathcal{D}' = \{x^{(j)}\}_{j=1}^{2N}$ is derived from the dataset $\mathcal{D} = \{(x_{\mathrm{L}}^{(i)}, x_{\mathrm{R}}^{(i)})\}_{i=1}^{N}$, where $x^{(2i-1)} = x_{\mathrm{L}}^{(i)}$ and $x^{(2i)} = x_{\mathrm{R}}^{(i)}$. The encoder model $q_\theta(x|z)$ and the decoder model $p_\theta(z|x)$ in the VAE are trained with the following optimization problem:

$$\max_{\theta} \mathop{\mathbb{E}}_{x\sim\mathcal{D}'} \left[\mathop{\mathbb{E}}_{z\sim q_\theta(z|x)} \left[\log p_\theta(x|z)\right] - KL\left(q_\theta(z|x)||p(z)\right) \right], \tag{1}$$

which is the maximization of the variational lower bound [9], the lower bound of the marginal log likelihood. Here, KL denotes the Kullback-Leibler (KL) divergence. Note that the sampling operation $z \sim q_\theta(z|x)$ is differentiable using the reparametrization trick. For the purpose of visualization in Fig.3, we name the two terms in the optimization problem $\mathcal{L}_{\mathrm{rec}} = -\mathbb{E}_{x\sim\mathcal{D}'}\left[\mathbb{E}_{z\sim q_\theta(z|x)}\left[\log p_\theta(x|z)\right]\right]$ and $\mathcal{L}_{\mathrm{pri}} = \mathbb{E}_{x\sim\mathcal{D}'}\left[KL\left(q_\theta(z|x)||p(z)\right)\right]$. For the rest of this paper, $\hat{\theta}$ denotes the estimated model parameter after the optimization of Eq.1. We use normal distribution for the encoder model $q_\theta(z|x)$.

2.5 Training Detail: Generator and Discriminator Training with Adversarial Learning

For notational simplicity, we introduce datasets of latent vectors $\mathcal{D}_z = \{(z_{\mathrm{L}}^{(i)}, z_{\mathrm{R}}^{(i)})\}_{i=1}^{N}$ and $\mathcal{D}'_z = \{z^{(j)}\}_{j=1}^{2N}$ corresponding to $\mathcal{D} = \{(x_{\mathrm{L}}^{(i)}, x_{\mathrm{R}}^{(i)})\}_{i=1}^{N}$ and $\mathcal{D}' = \{x^{(j)}\}_{j=1}^{2N}$ respectively, where $z_{\mathrm{L}}^{(i)} = \mathrm{argmax}_z\, q_{\hat{\theta}}(z|x_{\mathrm{L}}^{(i)})$, $z_{\mathrm{R}}^{(i)} = \mathrm{argmax}_z\, q_{\hat{\theta}}(z|x_{\mathrm{R}}^{(i)})$, and $z^{(j)} = \mathrm{argmax}_z\, q_{\hat{\theta}}(z|x^{(j)})$. The generator G and the discriminator D are parametrized by ψ and ϕ, respectively. In the following, we use short hand $\mathcal{L}^{\mathrm{p}}_{D_\phi}(z_{\mathrm{L}}, z_{\mathrm{R}}) = -\log D_\phi(z_{\mathrm{L}}, z_{\mathrm{R}})$, $\mathcal{L}^{\mathrm{n}}_{D_\phi}(z_{\mathrm{L}}, z_{\mathrm{R}}) = -(1 - \log D_\phi(z_{\mathrm{L}}, z_{\mathrm{R}}))$, and $\mathcal{U} = \mathcal{U}(0,1)^{d_s}$.

2.5.1 Discriminator Loss

In the adversarial learning framework, the discriminator classifies two sets of samples: the *real class* and the *fake class*. In Connective Fusion, samples in the real class are latent vector pairs $(z_{\mathrm{L}}, z_{\mathrm{R}})$ sampled from the dataset $\mathcal{D}_z$. Formally, the loss function for the real class of the discriminator is

$$\mathcal{L}^{\mathrm{p}}_{\mathrm{dis}} = \mathop{\mathbb{E}}_{(z_{\mathrm{L}}, z_{\mathrm{R}})\sim\mathcal{D}_z} \left[\mathcal{L}^{\mathrm{p}}_{D_\phi}(z_{\mathrm{L}}, z_{\mathrm{R}})\right]. \tag{2}$$

The fake class for the discriminator are latent vector pairs $(z_{\mathrm{L}}, z_{\mathrm{R}})$ which are obtained by i) sampling independently with the uniform distribution over the dataset $\mathcal{D}'_z$ (1st term of Eq.3), ii) sampling independently from the prior $p(z)$ (2nd term of Eq.3), iii) the generator G transforming samples $(z_{\mathrm{L}}, z_{\mathrm{R}}, s)$, where z_{L}, z_{R} are sampled in the same way as i, while s are sampled from $\mathcal{U}$, the uniform distribution over $\mathcal{S} = [0,1]^{d_s}$ (3rd term of Eq.3), and iv) the generator G transforming samples $(z_{\mathrm{L}}, z_{\mathrm{R}}, s)$, where z_{L}, z_{R} are sampled in the same way as ii, while s are sampled from $\mathcal{U}$ (4th term of Eq.3). Formally, the loss function is

$$\begin{aligned}\mathcal{L}^{\mathrm{n}}_{\mathrm{dis}} = &\mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim\mathcal{D}'_z}\left[\mathcal{L}^{\mathrm{n}}_{D_\phi}(z_{\mathrm{L}}, z_{\mathrm{R}})\right] \\ &+ \mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim p(z)}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim p(z)}\left[\mathcal{L}^{\mathrm{n}}_{D_\phi}(z_{\mathrm{L}}, z_{\mathrm{R}})\right] \\ &+ \mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}^{\mathrm{n}}_{D_\phi}(G_\psi(z_{\mathrm{L}}, z_{\mathrm{R}}, s))\right] \\ &+ \mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim p(z)}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim p(z)}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}^{\mathrm{n}}_{D_\phi}(G_\psi(z_{\mathrm{L}}, z_{\mathrm{R}}, s))\right].\end{aligned} \tag{3}$$

Finally, the overall loss function for the discriminator becomes

$$\mathcal{L}_{\mathrm{dis}} = \mathcal{L}^{\mathrm{p}}_{\mathrm{dis}} + \mathcal{L}^{\mathrm{n}}_{\mathrm{dis}}. \tag{4}$$

2.5.2 Generator Loss

Based on the discriminator, a generator is trained such that the generated samples become more like the real samples rather than the fake samples. We consider two kinds of generated samples which are sampled in the same way as iii and iv in the previous section 2.5.1. Formally, the loss function is

$$\begin{aligned}\mathcal{L}^{\mathrm{p}}_{\mathrm{gen}} = &\mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}^{\mathrm{p}}_{D_\phi}(G_\psi(z_{\mathrm{L}}, z_{\mathrm{R}}, s))\right] \\ &+ \mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim p(z)}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim p(z)}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}^{\mathrm{p}}_{D_\phi}(G_\psi(z_{\mathrm{L}}, z_{\mathrm{R}}, s))\right].\end{aligned} \tag{5}$$

Additionally, we introduce the similarity loss which is the latent space distance between samples before and after the transformation:

$$\begin{aligned}\mathcal{L}_{\mathrm{sim}} = &\mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim\mathcal{D}'_z}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}_{\mathrm{dist}}(z_{\mathrm{L}}, z_{\mathrm{R}}, s)\right] \\ &+ \mathop{\mathbb{E}}_{z_{\mathrm{L}}\sim p(z)}\mathop{\mathbb{E}}_{z_{\mathrm{R}}\sim p(z)}\mathop{\mathbb{E}}_{s\sim\mathcal{U}}\left[\mathcal{L}_{\mathrm{dist}}(z_{\mathrm{L}}, z_{\mathrm{R}}, s)\right],\end{aligned} \tag{6}$$

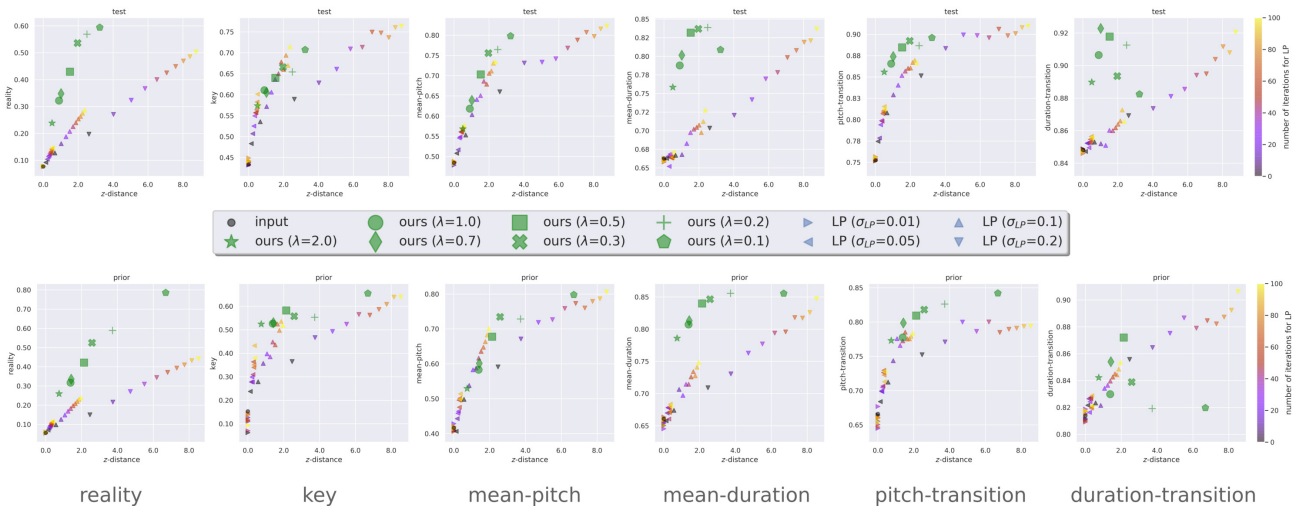


Figure 4: Evaluation of our model on six metrics vs z-distance. The upper/lower rows are transformation results where inputs are randomly created pairs of test/generated (sampled from prior) data. The color gradient corresponds to the number of iterations for LP. For all metrics, higher values are better. See Sec.3.5.

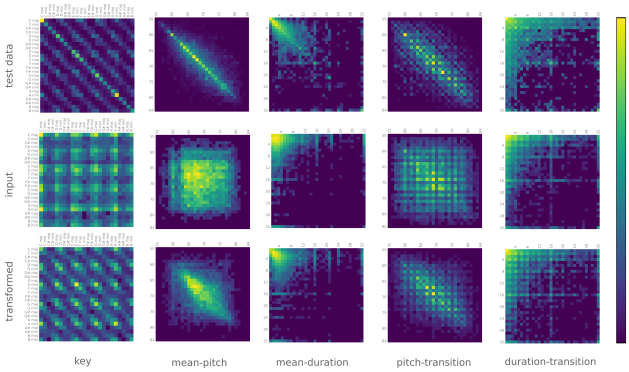


Figure 5: Musical statistics are corrected after our transformation. Brighter color indicates higher probability. See the second paragraph of 3.4.

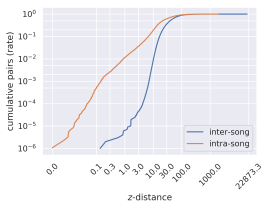


Figure 6: Analysis of z-distance in the latent space $\mathcal{Z}$. See the last paragraph of 3.4.

where

$$\mathcal{L}_{\text{dist}}(z_{\text{L}}, z_{\text{R}}, s) = \frac{1}{d_z}\left\|\frac{1}{\bar{\sigma}_z^2}\log\left(1+(z'_{\text{L}}-z_{\text{L}})^2\right)\right\|_1 + \frac{1}{d_z}\left\|\frac{1}{\bar{\sigma}_z^2}\log\left(1+(z'_{\text{R}}-z_{\text{R}})^2\right)\right\|_1$$
$$\text{with } (z'_{\text{L}}, z'_{\text{R}}) = G_\psi(z_{\text{L}}, z_{\text{R}}, s). \tag{7}$$

Following the latent constraint paper [6], $\bar{\sigma}_z \in \mathbb{R}^{d_z}$ is chosen to be the standard deviation $\sigma(x) \in \mathbb{R}^{d_z}$ of the encoder model $q_{\hat{\theta}}(z|x) = \mathcal{N}\left(\mu(x), \text{diag}\left(\sigma^2(x)\right)\right)$ averaged over all the training dataset. Precisely, $\bar{\sigma}_z = 1/|\mathcal{D}'|\sum_{x\in\mathcal{D}'}\sigma(x)$. Finally, the overall loss function for the generator becomes

$$\mathcal{L}_{\text{gen}} = \mathcal{L}^{\text{p}}_{\text{gen}} + \lambda\mathcal{L}_{\text{sim}}. \tag{8}$$

3. EXPERIMENTS

3.1 Dataset

We create datasets from LMD-matched of Lakh MIDI dataset [11], comprising 45,129 files matched to the song identity entries in the Million Song Dataset [12]. Each song has one or several different versions of MIDI files. We first extract files with 4/4 time signature, use accompanying tempo information to determine beat locations, and quantize each beat into 4. We then split the song identities into the proportion of 11:1:6:1:1 to create train-1, validation-1, train-2, validation-2, and test dataset, respectively. Train-1 and validation-1 datasets are for training proposed and baseline models, whereas train-2 and validation-2 datasets are for training evaluation models. We filter out non-monophonic tracks, bass or drum tracks, and the tracks outside the pitch range of [55, 84]. We conduct data augmentation by transposing tracks to all possible keys if the transposed tracks stay within the pitch range of [55, 84]. We retrieve 8-bar sliding windows (with a stride of 1 bar) from each track followed by filtering out windows that have more than one bar consecutive rests. Finally, for each split of train-1, validation-1, train-2, validation-2, and test dataset, we create a dataset $\mathcal{D} = \{(x_{\text{L}}^{(i)}, x_{\text{R}}^{(i)})\}_{i=1}^{N}$ by assigning the first 4-bars and the last 4-bars of each 8-bar window to $x_{\text{L}}^{(i)}$ and $x_{\text{R}}^{(i)}$, respectively. For encoding musical sequences, we use Melodico-rhythmic encoding proposed in [3].

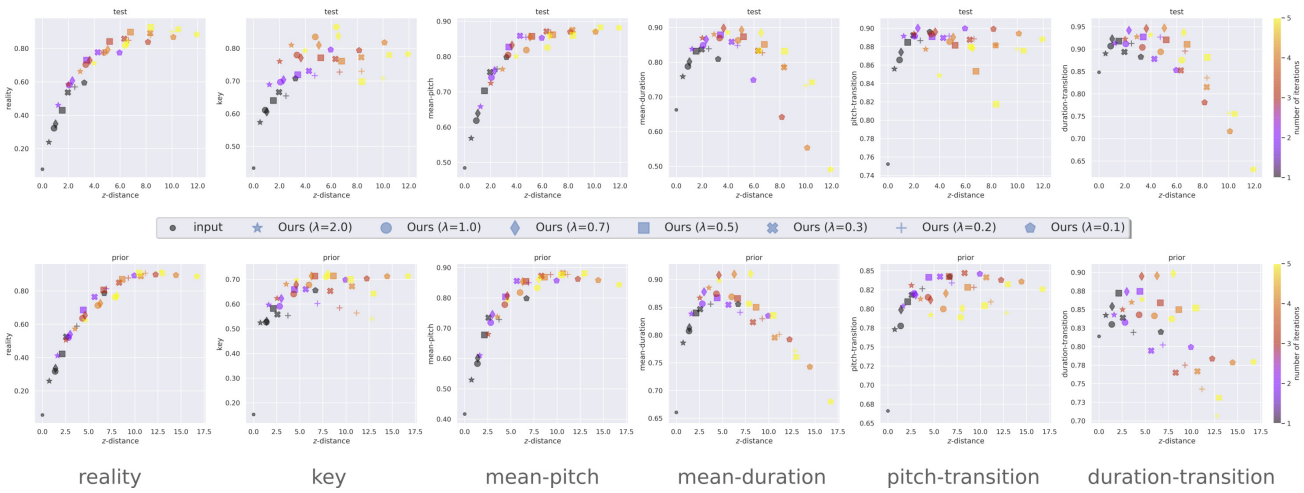


Figure 7: Evaluation of iterated application of our model. Iterated application of models with larger λ tends to compare favorably with others. The upper/lower rows are transformation results where inputs are randomly created pairs of test/generated (sampled from prior) data. The color gradient corresponds to the number of iterations for our method. For all metrics, higher values are better. See Sec.3.6.

3.2 Implementation Details

2-layered long short term memory (LSTM) [13] is used for the encoder and the decoder of the VAE, with the number of latent dimensionality $d_z = 64$. 5-layered multilayer perceptron (MLP) is used for the generator and the discriminator, where input vectors are simply concatenated and ReLU activation is employed. Following the latent constraint paper [6], we use the gating mechanism for each of the outputs z'_{L} and z'_{R}. For the decoder to generate sequences, argmax operation is utilized at each time step greedily for sampling each token. In the adversarial learning, the parameter updates of the model are alternating between updating the discriminator 10 times and updating the generator once. Adam optimizer [14] is used for training with the parameters $(\alpha, \beta_1, \beta_2) = (0.000005, 0, 0.9)$.

3.3 Transformation Examples

Fig.1 and Fig.2 are the transformation examples for $\lambda = 2.0$ and $\lambda = 0.3$, respectively. In Fig.1b at iteration 1, the used pitch set of x_{L} becomes aligned to that of x_{R}, and the note at time around 4 transforms to the one with longer duration to join two input sequences. As the number of iterations increases, the note durations become more uniform, while retaining similarity to the input sequences in terms of pitch contours or rhythmic properties. In Fig.2b, four corners are generated by feeding s randomly sampled from $\{0, 1\}$ for each dimension and the others are interpolations. Interestingly, the time 4 to 8 of the bottom left sequence is the exact transposition of the right input of Fig.2a, whereas other sequences has variations and transpositions.

3.4 Evaluation Metrics

We use *reality*, correlations of five musical statistics evaluated against the test data, and *z-distance* as evaluation metrics. Reality is defined as the probability that a generated sequence is classified as human-made. Two-layered transformer binary classification model is trained on train-2 and validation-2 datasets, where the pair sequences from the dataset $\mathcal{D}$ are labeled as positive, while the randomly shuffled pair sequences from the same dataset are labeled as negative. The mean of output values of this classification model and its accuracy are 0.959 and 0.953 on the test dataset. Reality is a holistic metric of quality which quantifies how likely a sequence is human-made.

As musical statistics, we choose to use key, mean-pitch (MP), mean-duration (MD) for evaluating whether the first and the last 4-bars are more musically consistent after transformation. We also employ pitch-transition (PT) and duration-transition (DT) for evaluating if transitions between the first and the last 4-bars are smooth and natural. Fig.5 illustrates these five musical statistics. For key, MP, and MD, these values are estimated for the first and the last 4-bars, which are interpreted as Markov transitions of musical states, creating matrices as in Fig.5. For PT and DT, we extract two or three consecutive notes around the boundary of the first and the last 4-bars, and again compute Markov transition matrices of each statistics as states. Finally, as a quantitative metric, we compute Pearson correlation coefficient between the matrix made from the test dataset and the one from the samples outputted by models.

z-distance is a scaled squared Mahalanobis distance $dist(z', z) = 1/d_z (z' - z)\Sigma^{-1}(z' - z)^{\intercal}$, where $\Sigma = \mathrm{diag}(\bar{\sigma}_z^2)$. We analyze z-distance in the latent space $\mathcal{Z}$ using song id annotations of the dataset. We randomly sampled 1,000,000 pairs of 4-bar sequences from the entire dataset, where pairs are sampled either from the same or different song id, corresponding to intra- and inter-song in Fig.6. Pairs with identical sequences are filtered out to analyze the similarity rather than the identicality. Fig.6 shows

that sequences from different song ids are several orders of magnitude less likely to have smaller (say, less than 1.0 or 3.0) z-distances, compared with those from the same ids. The difference of z-distances probably comes from the fact that a lot of variations can be seen within each song id and similar themes are rarely seen among different song ids especially for melody, suggesting that smaller z-distances includes variations and that z-distance encodes musically meaningful similarity.

Algorithm 1 Latent Perturb (baseline method).

Input: z_{L}, z_{R}, and σ_{LP}

$S_{\max} \leftarrow \mathrm{argmax}_x\, p_{\hat{\theta}}(x|z_{\mathrm{L}}) \oplus \mathrm{argmax}_x\, p_{\hat{\theta}}(x|z_{\mathrm{R}})$
$R_{\max} \leftarrow$ estimate reality of $S_{\max}$
for $i = 1$ to $NumIter$ **do**
  $\epsilon_{\mathrm{L}}, \epsilon_{\mathrm{R}} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\sigma^2_{\mathrm{LP}}))$
  $S_{\mathrm{L}} \leftarrow \mathrm{argmax}_x\, p_{\hat{\theta}}(x|z_{\mathrm{L}} + \epsilon_{\mathrm{L}})$
  $S_{\mathrm{R}} \leftarrow \mathrm{argmax}_x\, p_{\hat{\theta}}(x|z_{\mathrm{R}} + \epsilon_{\mathrm{R}})$
  $R_{\mathrm{tmp}} \leftarrow$ estimate reality of $S_{\mathrm{tmp}} = S_{\mathrm{L}} \oplus S_{\mathrm{R}}$
  if $R_{\mathrm{tmp}} > R_{\max}$ **then**
    $S_{\max} \leftarrow S_{\mathrm{tmp}};\ R_{\max} \leftarrow R_{\mathrm{tmp}}$
    $z_{\mathrm{L}} \leftarrow z_{\mathrm{L}} + \epsilon_{\mathrm{L}};\ z_{\mathrm{R}} \leftarrow z_{\mathrm{R}} + \epsilon_{\mathrm{R}}$
  end if
end for
return $S_{\max}$

3.5 Comparisons of Methods

We introduce a naive but strong baseline method called *Latent Perturb (LP)*, which is summarized in Algorithm 1. Note that the reality estimation model is trained with the train-1 dataset, and the argmax operation is approximated with the greedy sampling scheme. Compared to data space, perturbation in the latent space can efficiently search similar samples with high reality.

Fig.4 shows the comparisons of baseline methods and ours evaluated on z-distance vs the other evaluation metrics, where z-distance is the average of the mean of z-distances for z_{L} and z_{R}. The reason for choosing LP with σ_{LP} in $[0.01, 0.2]$ is to make sure their resulting evaluation metrics span the range that the proposed technique "ours" spans. For samples from the prior as well as from the test dataset, our methods, especially with larger λ tend to outperform in all the metrics, even though LP methods use abundant computational budgets—100 forward computations of the decoder model and the classification model.

3.6 Evaluation of Iterated Transformation

Fig.7 illustrates the performance of iterated application of our transformation. Interestingly, performance of models with larger λ after several iterations are often comparable to or better than that of models with smaller λ. This means that a model with larger λ can be used for transformation with different distances, without sacrificing the performance. For example, as illustrated in Fig.1, the model with $\lambda = 2.0$, if iteratively applied, can be used for gradual transformations with several different distances. Their quality is as satisfactory as single (non-iterative) transformation with models of smaller λ.

4. RELATED WORK

Concatenative Synthesis (CS) methods [15, 16] typically use a large database of source audio segmented into units, and search units that match each segment of the target audio by unit selection algorithms. Mashup methods [17, 18] combine two or more songs to create entertaining musical results. Typically tracks from different songs are superimposed and combined. Our Connective Fusion differs from CS and Mashup in that i) ours is generative rather than searching and using existing segments of music, ii) ours tries to combine any segments, whereas CS/Mashup typically combine some searched segments, iii) ours is unsupervised learning which does not need any human annotation in the dataset or musical expertise for training models, and iv) ours is symbolic which is often not the case in CS/Mashup.

Learning transformation in the latent space has been studied for image, texts, and music [6, 19–23]. Notably, Mueller et al. proposed to transform texts to have specified attributes based on optimization in the latent space [21]. Engel et al. proposed to transform image or music to become more realistic or to have specified attributes in the latent space by using adversarial learning [6]. Lore et al. and more recently Jahanian et al. instead proposed to transform image by learning latent space directions [22, 23].

Learning transformation with adversarial learning without paired data has been extensively studied. The input and output domain of transformation in these studies can be categorized into the same data domain with different classes [24] or attributes [6,25], the semantic domain to the data domain [24], and semantically similar domain such as simulation to real [26, 27] and unsupervised language translation [28, 29]. On the other hand, the input and output domains of transformation in this paper are essentially the same except that they have different lengths/counts.

5. CONCLUSION

We introduced Connective Fusion, a new scheme of interactively generating sequences for creative purposes. Given two sequences of random combination, our model transforms each of them to similar one such that their concatenation is more indistinguishable from human-made sequences. The transformation model is adversarially learned in the latent space. Our model equips with user control—choosing the amount of transformation as well as exploring in the style space. In experiments of melody creation on a symbolic music dataset, we empirically demonstrated that our method outperforms a baseline method of various parameter settings, and that iterative gradual transformation not just introduced new functionality but also works as satisfactory as or sometimes better than non-iterative transformation.

6. REFERENCES

[1] 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: Generating music with long-term structure," in *International Conference on Learning Representations*, 2019.

[2] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proc. of the 34th International Conference on Machine Learning*, 2017.

[3] 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. [Online]. Available: https://doi.org/10.1007/s00521-018-3868-4

[4] J. Gillick, A. Roberts, J. H. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, 2019.

[5] 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.

[6] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," in *International Conference on Learning Representations*, 2018.

[7] 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.

[8] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014.

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

[12] 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.

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

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

[15] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 1996, pp. 373–376 vol. 1.

[16] A. Zils and F. Pachet, "Musical Mosaicing," in *Proceedings of the International Conference on Digital Audio Effects*, 2001.

[17] N. Tokui, "Massh! - a web-based collective music mashup system," in *Proceedings - 3rd International Conference on Digital Interactive Media in Entertainment and Arts, DIMEA 2008*, 2008.

[18] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "Automashupper: Automatic creation of multi-song music mashups," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 2014.

[19] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems 29*, 2016.

[20] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] J. Mueller, D. Gifford, and T. Jaakkola, "Sequence to better sequence: Continuous revision of combinatorial structures," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[22] L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola, "Ganalyze: Toward visual definitions of cognitive image properties," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[23] A. Jahanian*, L. Chai*, and P. Isola, "On the "steerability" of generative adversarial networks," in *International Conference on Learning Representations*, 2020.

[24] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision, ICCV*, 2017.

[25] G. Lample, S. Subramanian, E. Smith, L. Denoyer, M. Ranzato, and Y.-L. Boureau, "Multiple-attribute text rewriting," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1g2NhC5KQ

[26] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

[27] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

[28] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised machine translation using monolingual corpora only," in *International Conference on Learning Representations*, 2018.

[29] M. Artetxe, G. Labaka, E. Agirre, and K. Cho, "Unsupervised neural machine translation," in *International Conference on Learning Representations*, 2018.