SDMuse: Stochastic Differential Music Editing and Generation via Hybrid Representation

Chen Zhang®, Yi Ren, Kejun Zhang®, and Shuicheng Yan®, Fellow, IEEE

Abstract—While deep generative models have empowered music generation, it remains a challenging and under-explored problem to edit an existing musical piece at fine granularity. In this article, we propose SDMuse, a unified Stochastic Differential Music editing and generation framework, which can not only compose a whole musical piece from scratch, but also modify existing musical pieces in many ways, such as combination, continuation, inpainting, and style transferring. The proposed SDMuse follows a two-stage pipeline to achieve music generation and editing on top of a hybrid representation including pianoroll and MIDI-event. In particular, SDMuse first generates/edits pianoroll by iteratively denoising through a stochastic differential equation (SDE) based on a diffusion model generative prior, and then refines the generated pianoroll and predicts MIDI-event tokens auto-regressively. We evaluate the generated music of our method on aildabs1k7 pop music dataset in terms of quality and controllability on various music editing and generation tasks. Experimental results demonstrate the effectiveness of our proposed stochastic differential music editing and generation process, as well as the hybrid representations.

Index Terms—Music editing, music generation, stochastic differential equation, generative models.

I. INTRODUCTION

With the development of deep learning and generative models, automatic music composition has received much research attention [6], [11], [12], [18], [24], [25], [28], and also has a lot of successful applications, such as movie soundtracks, virtual singers, and auxiliary composing. However, as the aesthetics of music are diverse for different groups of people, or even for each individual, there is no musical piece in the world that simultaneously satisfies everyone’s taste for any scenario. In many cases, one may feel unsatisfied with certain bars of a musical piece, if not with the whole piece, and he or she would like to edit them, which is impossible except for a professional such as a music composer. AI solutions like symbolic music generation methods [10], [11] can help, but they would regenerate a completely new piece that may be totally different from the former one without preserving those “satisfying” parts. The same thing happens when someone wants to extend an existing musical piece, modify the style or combine several pieces. Therefore, besides the effort on improving generative quality [12], [18] in a broad sense, it is also crucial and much demanded to study how to edit and modify existing musical pieces.

There are only a few works studying music editing tasks, mainly focusing on global music style transfer [2], [5], [25] and music inpainting [8]. The most related work [25] to ours achieves music style transfer with Transformer VAE on pre-defined musical attributes like rhythmic intensity and polyphony. The musical attributes are song-level, prohibiting users from editing certain part of a musical piece. In addition, users can only control and edit the pre-defined attributes, which severely limits its application. Other works [2], [5] have similar limitations. [8] only conduct MIDI inpainting with fixed length, which makes it not flexible for users. So far, no one has explored editing musical pieces at fine granularity in the same flexible way as image editing, such as stroke-based generation and editing.

Existing state-of-the-art music generation works [10], [18], [28] with high generative quality are based on MIDI-event representation (an illustration of MIDI-event is in Fig. 1(c)). As validated by the success of these methods, the MIDI-event representation is appropriate for the model to generate and model music performance details such as velocity, but not for humans to perceive and edit since the data format is not intuitive enough. Another widely used music representation pianoroll (an illustration of pianoroll is in Fig. 1(b)) is closer to the way of human perception, but the methods [6], [26] based on it cannot achieve comparable generation quality with MIDI-event-based ones. In this work, we consider using both of the representations, which we call hybrid representation, to achieve fine-grained music editing.

Specifically, we define a series of fine-grained music editing tasks and propose a unified Stochastic Differential Music editing and generation framework via hybrid representation, named SDMuse. SDMuse can not only generate a whole musical piece from scratch either unconditionally or conditioned on some control signals (such as chord progression) but also edit existing musical pieces in different ways, including stroke-based generation/editing, inpainting, outpainting, combination and style transfer. As pianoroll is easier for humans to understand and
edit, and MIDI-event sequence is more suitable for generation, we design a two-stage pipeline based on a hybrid representation. In the first stage, a diffusion model generative prior is applied and the pianorolls are obtained by iterative denoising through a stochastic differential equation (SDE). The progressive generation feature of our diffusion model allows us to implement a series of music editing operations in this stage. In the second stage, the generated pianoroll will be refined with more precise music performance details (velocity, fine-grained onset position, etc.) by generating the final MIDI-event sequence in an auto-regressive manner, enjoying the benefits of MIDI-event representation for high-quality music generation.

We evaluate SDMuse on ailabs1k7 pop music dataset [10] in terms of quality and controllability in various music editing and generation tasks. Both objective and subjective results demonstrate the effectiveness of the stochastic differential process and hybrid representation in SDMuse. To our best knowledge, we are the first one to formulate and address fine-grained music editing, which aims to edit existing musical pieces at fine granularity according to diverse demands and provide humans with new ways to collaborate with models during music composition. The generated samples can be found on our demo page https://sdmuse.github.io/posts/sdmuse/.

II. BACKGROUND

A. Symbolic Music Representation

Most previous symbolic music generation works are based on two most common symbolic music representations: pianoroll and MIDI-event. Pianoroll-based approaches [6], [26] use pianorolls to represent music scores, with the horizontal axis representing time and the vertical axis representing pitch. Considering that pianoroll is just like an image, pianoroll-based methods use image-based operations to generate music. Pianoroll is closer to the way of humans perceiving musical pieces, making it more suitable for being understood and edited by humans. However, most state-of-the-art music generation works are MIDI-event-based approaches [10], [11], [17]. They convert a musical piece to a MIDI-event token sequence and use methods from natural language processing to deal with the token sequence. MIDI-event-based methods can better learn the temporal dependency between different musical events, so as to show a more robust generation performance. [15] used diffusion models to generate symbolic music, converting discrete MIDI-event token sequences to continuous latent by pretrained MusicVAE [19]. However, it is simply stitching of the diffusion model and MusicVAE, thus the generated performance is limited by the performance of the pretrained MusicVAE model.

As listed in Table I, MIDI-event and pianoroll have their own advantages and disadvantages when representing a piece of symbolic music. Pianoroll contains more prior music information. Pitch, duration, and relative position of notes are directly shown on it, so one can easily perceive the music structure, note density, etc. of an entire song when given a pianoroll, which makes it more suitable to be edited by humans. However, there is no difference in the way that a pianoroll treats the onset and other positions of a note, which is not consistent with real music. On the other hand, MIDI-event sequence can carry more precise details than pianoroll, such as velocity, which can enhance the richness and expressiveness of the generated music performance. Furthermore, MIDI-event regards a note as the unit and is more robust to generate musical pieces. When converting the MIDI-event tokens into embeddings whose weights are randomly initialized, the correlations between different MIDI-events are lost and need to be relearned from the training data. Given the circumstance that the amount of high-quality symbolic music training data is limited, it is relatively easier for a deep learning model to extract musical information from pianoroll than MIDI-event.

To summarize, the pianoroll is more appropriate for extracting and controlling perceptive information like structure, while the MIDI-event sequence is ideal for generating and modeling precise music performance details, such as velocity and fine-grained onset position. In this work, we propose SDMuse with a two-stage pipeline, including pianoroll and MIDI-event generation stages that apply hybrid representation to take advantage of these two symbolic music representations.

B. Symbolic Music Editing

Though tremendous progress is made in symbolic music generation and other automatic music composition tasks, there are only a handful of works regarding symbolic music editing tasks. The existing symbolic music editing works mainly focus on global music style transfer [2], [5], [25] and music inpainting [3], [8], [5] proposed a system for music accompaniment style transfer, generating accompaniment with the content from content input and the style from style input. [25] changed the style of a musical piece according to the given song-level musical attributes (e.g. rhythmic intensity and polyphony). [2]
applied GAN-based methods from computer vision to transfer a musical piece from the source genre to the target genre. However, these works can only edit the music at the song level with pre-defined attributes or labels, limiting the ways of interaction. [8] proposed a Piano Inpainting Application plugin for interactive and responsive A.I.-assisted composition by using Linear Transformer, while cannot conduct MIDI inpainting with flexible length. [3] addressed the problem of variable-length MIDI infilling by proposing a new positional encoding method, but the infilling content remains uncontrollable. In this article, SD-Muse aims to implement fine-grained music editing and achieve flexible collaboration between humans and models during music composition.

C. Stochastic Differential Equations (SDE) for Editing

To recover the data from noise, [21] proposed a stochastic differential equation (SDE) to smoothly transform a complex data distribution to a known prior distribution by slowly injecting noise. Similar to diffusion probabilistic models [9], [13], [15], SDE-based generative models can be used to convert an initial Gaussian noise vector to a data point in real-world data distribution. As described in [21], we denote $x(t) \in \mathbb{R}^d$, where $t \in [0, T]$ represents time. Suppose that $x(0) \sim p_{data}$ is a sample from data distribution, and $x(T) \sim p_T$ is from the prior distribution. The forward SDE process can be formulated as:

$$dx(t) = f(x, t)dt + g(t)dw, \quad (1)$$

where $w$ is a standard Brownian motion, and $f(\cdot, t), g(\cdot)$ are the drift coefficient and the diffusion coefficient of $x(t)$ respectively. And the reverse SDE [1] is:

$$dx(t) = [f(x, t) - g(t)^2\nabla_x \log p_T(x)]dt + g(t)d\bar{w}, \quad (2)$$

where $\bar{w}$ is a standard Wiener process and $\nabla_x \log p_T(x)$ is the noise-perturbed score function. There are two different SDEs according to different noise perturbations: Variance Exploding SDE (VE-SDE) and Variance Preserving SDE (VP-SDE).

In this article, we use VP-SDE for conducting experiments to verify our framework. The forward process of VP-SDE is:

$$dx(t) = -\frac{1}{2} \beta(t)x(t)dt + \sqrt{\beta(t)}dw(t), \quad (3)$$

where $\beta(t)$ is a positive noise function. Denote the learned score model as $s_\theta(x(t), t))$. The discretized version of the reverse VP-SDE process can be solved by following the iteration rule:

$$x_{n-1} = \frac{1}{\sqrt{1 - \beta(t_n)\Delta t}} (x_n + \beta(t_n)\Delta ts_\theta(x(t_n), t_n)) + \sqrt{\beta(t_n)\Delta t} z_n,$$

where $x_N, z_n \sim N(0, I), n = N, N - 1, \ldots, 1$, and $\Delta t$ is time interval between $x_n$ and $x_{n-1}$.

In order to synthesize and edit images, [14] “hijacked” the generative process of SDE-based generative models. Specifically, they added noise to smooth the given stroke paintings or images with stroke-edit to smooth out distortions while preserving the overall image structure. Then they used the noisy input to initialize the SDE and progressively remove the noise. Inspired by [14], we build our SDMuse to edit and generate musical pieces through VP-SDE.

III. METHODS

In this section, we first give a pipeline overview of SDMuse and then describe the design details of the pianoroll and MIDI-event generation stages respectively. Finally, we introduce the formulation and process of various fine-grained music editing tasks.

A. Pipeline Overview

The overall pipeline of SDMuse is shown in Fig. 2(a), which consists of two consecutive stages: 1) pianoroll generation stage, which is based on a conditioned diffusion model generative prior and synthesizes pianorolls from scratch or edits existing pianorolls by iteratively denoising through SDE [21]; 2) MIDI-event generation stage, which generates MIDI-event sequences from the output pianorolls of the first stage by refining them with precise music performance details auto-regressively. These two stages can be trained separately and all condition signals in the first stage are at fine granularity, which can be extracted from the musical piece itself without extra data annotation process. The diffusion probabilistic model in the pianoroll generation stage enables SDMuse not only to compose the whole musical pieces from scratch unconditionally or conditioned on given control signals (e.g. note density), but also to modify existing musical pieces in different ways. The details of these two stages are introduced in the following subsections and some details of the model architectures are described in the supplementary materials.
B. Pianoroll Generation Stage

1) Training of Diffusion Probabilistic Model: As shown in Fig. 2, we involve several fine-grained control signals: note density $c_n$, pitch distribution $c_p$, and chord progression sequence $c_c$ during the training process of the diffusion model to enable unconditional and conditional music generation/editing at the same time. These control signals can be extracted from the musical piece itself, and the way of extraction is listed in the supplementary materials. Given these control signals $c_n$, $c_p$, and $c_c$, we can train a conditional diffusion model with the pairs of pianorolls and the corresponding control signals. In order to integrate unconditional and conditional music generation into the same model without an extra training process, we introduce a combined training strategy which can switch freely between two generation settings inspired by [16]. Specifically, as illustrated in Fig. 2(a), we use the conditional training paradigm, but assign all control signals to a specific out-domain value ($c_{null}$) with a certain probability $p_{uncond}$ to train the model for unconditional music generation scenario. As mentioned in Section II-A, one of the drawbacks of pianoroll representation is indiscriminate treatment of note onsets and other positions, which does not match realistic scenarios and affects the robustness of generation. To tackle this problem, we convert the pianoroll to onsetroll by only keeping onset information as described in the supplementary materials for the training process of the diffusion model.

2) Generation From Scratch: Starting from samples of $x(T) \sim p_T$, where $p_T$ is the prior distribution, we can generate musical pieces of $x(0) \sim p_{data}$ unconditionally or conditioned by given control signals. That is to say, we are interested in the $p(x|c)$, where

$$
    c = \begin{cases} 
    c_{null}, & \text{unconditional generation,} \\
    \{c_n, c_p, c_c\}, & \text{conditional generation.} 
    \end{cases}
$$

Derived from the reverse-time SDE in Equation 2, the conditional reverse-time SDE can be described:

$$
    dx(t) = [f(x, t) - g(t)^2 \nabla_x \log p_t(x|c)]dt + g(t)d\mathbf{w},
$$

where $\mathbf{w}$ is a standard Wiener process, $\nabla_x \log p_t(x|c)$ is the conditioned noise-perturbed score function, $f(\cdot, t)$ and $g(\cdot)$ are the drift coefficient and diffusion coefficient of $x(t)$ respectively. Thus, by given different $c$ for the reverse SDE process, the pianoroll generation stage in SDMuse can achieve unconditional music generation and conditional music generation respectively.

3) Fine-Grained Editing: Similar to [14], the diffusion probabilistic model can serve various fine-grained music editing tasks. We formulate and introduce the following fine-grained music editing tasks, which are illustrated in Fig. 2(b) from top to bottom respectively. All of the following tasks can be implemented with the same algorithm framework (Algorithm 1) with different process operations to obtain the input pianoroll $x$ and mask $\Omega$ and different reverse steps $t_0$. The mask $\Omega$ demonstrates the regions that need to be reserved all the time by setting the value to 0 and the regions that need to be replaced by setting the value to 1. Besides the style transfer which requires control
Simulated fine-grained music editing (VP-SDE) can be used in both unconditional and conditional settings.

**Stroke-based generation:** Like stroke-based image generation mentioned in [14], stroke-based generation aims to generate the realistic pianoroll from the given stroke pianoroll. The generated pianorolls are expected to balance between faithfulness and realization, which means that they should not only be similar to the given stroke pianoroll but also be authentic and reasonable. One can draw a stroke pianoroll with a specific structure, thus enabling structure-conditioned music generation. The mask $\Omega$ is set to 1 for all regions and the reverse steps $t_0$ is set to 0.4.

**Stroke-based editing:** When someone is unsatisfied with an existing pianoroll and wants to edit a certain part of it, the stroke-based editing can help. Given a pianoroll with stroke edits, we can generate a realistic pianoroll that follows the editing information, keeping the other parts from being changed. Stroke-based editing allows users to polish a given pianoroll to their liking, which is useful for eliminating bad cases and personalized music generation. The mask $\Omega$ is set to 1 for the edited regions and reverse steps $t_0$ is set to 0.4.

**Inpainting/outpainting:** Similar to image inpainting [27] and outpainting [23], we would like to reconstruct missing regions or extend the border of existing pianoroll. Inpainting can be used for music detail filling and outpainting can be used for music continuation, which is important to generate music pieces with flexible lengths. For inpainting, the mask $\Omega$ is set to 1 for the missing regions and the reverse steps $t_0$ is set to 1. And for outpainting, we concatenate the pianoroll and a random initialized part as input $x$ and set the mask $\Omega$ to 1 only for the randomly initialized regions and the reverse steps $t_0$ to 1.

**Combination:** Another important scenario is combining several music segments together harmoniously, which can be applied when someone likes several segments and wants them to appear in the same musical piece. We concatenate the pieces with some parts which are sampled from the prior distribution $p_r$, and use this as input $x$. Similar to outpainting, the mask $\Omega$ is set to 1 only for the randomly initialized regions and the number of reverse steps $t_0$ is set to 1.

**Style transfer:** As described in [25], given an existing musical piece $x$, we can change it to another style by adjusting some control signals such as rhythmic intensity, note density, pitch distribution, etc. For example, if the note density of a certain music piece increases, the music piece will sound more intense or upbeat. We can achieve precise local control cause control signals in SDMuse are fine-grained. The mask $\Omega$ is set to 1 for all regions and $t_0$ is set to 0.4.

### C. MIDI-Event Generation Stage

The MIDI-event generation stage is designed for refining the generated pianorolls from the prior stage with more precise music performance details by generating the final MIDI-event sequences auto-regressively, thus being able to benefit from the advantages of MIDI-event representation for robust music generation. As shown in the upper part of Fig. 2(a), this stage consists of 1) a score encoder to encode music score (pianoroll generated by the prior stage) into score condition; 2) a “expand by bar” option to add position information and expand the score condition to align with the decoder input; and 3) an auto-regressive decoder to generate MIDI-event tokens step by step.

Specifically, the score encoder (see Fig. 1(a) in supplementary materials) is a 12-layer convolution network with group normalization, which takes the music score (pianoroll) as input to generate the corresponding score condition. The score condition is then concatenated with barpos embedding, which is used to indicate the position and introduced by [18], and tiled to the same length with decoder input (MIDI-event token sequence) according to the bar information from decoder input. We call the set of these operations as “expand by bar” and illustrate it in Fig. 1(b) of the supplementary materials. Finally, the expanded
score condition can be added to the decoder input and forwarded to the auto-regressive decoder. The auto-regressive decoder is a transformer decoder [22], which can also be regarded as a conditional language model, helping with eliminating outliers predicted in pianorolls and adding more precise music performance details.

IV. EXPERIMENTS

In this section, we first introduce the experimental setup including dataset, evaluation metrics, baselines, etc. Then we report the results of unconditioned and conditioned generation with SDMuse. And finally, we show the performance of SDMuse in various fine-grained music editing tasks with evaluation results and cases. The audio samples can be found in our demo page.1

A. Experimental Settings

Datasets: In our experiments, we use the ailabs1k7 dataset introduced by [10], which contains 1,748 pieces of pop piano performance. We process all the pieces in training set into 32-bar segments by sliding window, with window size of 32 bar and hopping size of 4 bar, thus obtaining around 15,000 segments for the training of conditioned diffusion model in pianoroll generation stage and encoder-decoder in MIDI-event generation stage. For pianoroll, we set the granularity to 1 beat, which means the length of pianoroll n is the beat number of the corresponding musical piece. And for MIDI-event sequence, we follow the representation in [18], ignoring the track and instrument information because the music of ailabs1k7 dataset is single-track polyphonic music.

Evaluation metrics: We evaluate our results based on quality and controllability, for which we performed both objective and subjective assessments. As listed in Table II, to evaluate quality, we use PD and DD scores introduced in [20] and conduct subjective evaluation to obtain the overall perceptive scores. On the other hand, for quantifying controllability of conditioned music generation and editing, we calculate the L2 distance of control signals (CSD) between the given one and those of generated output. Also, when conducting fine-grained music editing like stroke-based generation, we compute the overlap ratio (OR) between the generated pianorolls and the input stroke pianorolls. And we evaluate the consistency of the edited samples in fine-grained music editing subjectively. Similar to [7], [28], we invite 10 participants with music knowledge to give their scores (five-point scales, 1 for bad and 5 for excellent) of randomly selected samples. The detailed instruction given to annotators is posted in the supplementary materials.

Baselines: For comparison, we choose different types of symbolic music generation and style transfer models as our baselines: 1) REMI [11]; 2) CPW [10] and 3) MuseMorphose [25]. We use the official implementation of each model and train these three models with the same training set. Considering that we are the first to conduct fine-grained music editing tasks, we only compare the quality of generated outputs with these baselines.

Model configuration: For the pianoroll generation stage, we use Gaussian diffusion model2 and adjust the UNet architecture [4] to make sure it can take control signals as the condition. And for the MIDI-event generation stage, we use a 4-layer transformer decoder and a 12-layer convolution 1D encoder. Other details about the model hyper-parameters are listed in the supplementary materials.

Training setup: The training data of both modules is cut into 32-bar segments. We train the diffusion model with diffusion step of 100 and use the linear noise schedule with max beta of 0.02. The training process of the pianoroll generation stage takes about 2 days on 1 A100 GPU with batch size of 32 pianorolls. And the MIDI-event generation stage is trained for around 12 hours on 1 A100 GPU with batch size of 2000 MIDI-event tokens.

B. Generation From Scratch

1) Unconditioned Generation: We first evaluate the performance of SDMuse on the unconditioned music generation task by just setting the control signals c as cnull in the pianoroll generation stage. As shown in Table III, denoted as SDMuse (unconditioned), while unconditioned generation is not our primary goal, we find that SDMuse achieves comparable results to the state-of-the-art music generation models, which indicates the effectiveness of our training strategy of diffusion probabilistic model that switches between unconditioned and conditioned generation settings. We present qualitative comparison results in Fig. 3 by just showing the pianorolls extracted from the final outputs of SDMuse and baselines.

2) Conditioned Generation: In order to assess the performance of SDMuse when generating musical pieces from scratch conditioned on given control signals, we use the control signals extracted from the test set during the pianoroll generation stage. The quality results are also listed in Table III, denoted as SDMuse (conditioned), demonstrating that with the guidance of

| Objective | Quality | Controllability |
|-----------|---------|-----------------|
| pitch distribution similarity (PD) | control signal distance (CSD) |
| duration distribution similarity (DD) | overlap ratio (OR) |

1[Online]. Available: https://SDMuse.github.io/posts/sdmuse/
2[Online]. Available: https://github.com/openai/guided-diffusion
TABLE III
OBJECTIVE AND SUBJECTIVE RESULTS OF BASELINE SYSTEMS IN MUSIC GENERATION, AND SDMUSE IN GENERATION FROM SCRATCH TASKS (BOTH UNCONDITIONED AND CONDITIONED ON GIVEN CONTROL SIGNALS) AND FINE-GRAINED MUSIC EDITING TASKS IN TERMS OF QUALITY

| Task       | Model / Setting  | Objective | Subjective        |
|------------|------------------|-----------|-------------------|
|            |                  | PD ↑  | DD ↑  | overall perceptive score ↑ |
| GT         |                  | -     | -     | 4.07 (±0.09)       |
| Generation | REMI [11]        | 0.82  | 0.76  | 3.52 (±0.07)       |
|           | CPW [10]         | 0.74  | 0.80  | 3.71 (±0.06)       |
|           | SDMuse (unconditioned) | 0.84  | 0.81  | 3.69 (±0.06)       |
|           | SDMuse (conditioned) | 0.88  | 0.79  | 3.72 (±0.06)       |
| Editing    | MuseMorphose [25]* | 0.68  | 0.81  | 3.80 (±0.06)       |
|           | SDMuse (stroke-based generation) | 0.84  | 0.80  | 3.47 (±0.07)       |
|           | SDMuse (stroke-based editing)* | 0.96  | 0.88  | 3.81 (±0.07)       |
|           | SDMuse (inpainting)* | 0.96  | 0.87  | 3.70 (±0.06)       |
|           | SDMuse (outpainting) | 0.79  | 0.75  | 3.63 (±0.07)       |
|           | SDMuse (combination) | 0.86  | 0.83  | 3.59 (±0.09)       |
|           | SDMuse (style transfer)* | 0.92  | 0.80  | 3.77 (±0.07)       |

Settings with * notation have high PD, DD, and subjective perceptive scores because they are based on existing musical pieces with only minor edits. The overall perceptive scores are calculated with 95% confidence intervals.

Fig. 3. The pianorolls extracted from the music generated by SDMuse and baseline systems. We randomly post three output pianorolls for each system.

Fig. 4. Pianorolls (b) generated by the pianoroll generation stage and (c) extracted from the final output music generated by the MIDI-event generation stage in stroke-based generation task. Figure (b) illustrates that the pianoroll generation stage can produce pianorolls based on the input stroke, although there may be some outliers. Figure (c) indicates that the MIDI-event generation stage can refine the pianoroll output from the previous stage while retaining the main structure and notes.

explicit music information from control signals, SDMuse can obtain better generation quality with reasonable listening experience compared to unconditioned generation. And the controllability results are presented in Table IV, indicating that SDMuse has the ability to generate musical pieces based on the control signals faithfully. Also, as a complement, we compare the note density $c_n$ and the pitch distribution $c_p$ extracted from output music with the given ones, illustrated in Fig. 5(c), for a visual demonstration of the faithfulness in terms of control signals during conditioned generation.
TABLE IV
OBJECTIVE AND SUBJECTIVE RESULTS OF STYLE TRANSFER BASELINE AND SDMUSE IN CONDITIONED MUSIC GENERATION TASK AND VARIOUS FINE-GRAINED MUSIC EDITING TASKS IN TERMS OF CONTROLLABILITY

| Model / Setting                        | Objective | Subjective          |
|---------------------------------------|-----------|---------------------|
|                                       | CSD (τn) ↓ | CSD (τp) ↓ | OR ↑ | overall consistency score ↑ |
| MuseMorphose [25]                     |           | -                   | -    | 3.87 (±0.07)                |
| SDMuse (conditioned)                  | 0.06      | 0.15                | 0.85 | 3.43 (±0.10)                |
| SDMuse (stroke-based generation)      | -         | -                   | 0.81 | 3.89 (±0.08)                |
| SDMuse (stroke-based editing)         | -         | -                   | 0.81 | 3.89 (±0.08)                |
| SDMuse (style transfer)               | 0.12      | 0.38                | 0.85 | 4.02 (±0.06)                |

The subjective scores are calculated with 95% confidence intervals.

Fig. 5. Visualization of the difference in c_n and c_p with different control signal’s embedding ways.

C. Fine-Grained Editing

For the aforementioned fine-grained editing tasks (see Section III-B3 for details), we edit existing musical pieces in corresponding ways and evaluate these tasks respectively. The quality results are shown in Table III and the controllability results are shown in Table IV. It is obvious that the tasks with only minor edits, such as stroke-based editing, inpainting and style transfer, perform well in both quality and controllability. For the stroke-based generation which highly depends on the given stroke pianorolls, there is a trade-off between the PD/DD scores and OR score. Fig. 4 illustrates the process of stroke-based generation task, including the input stroke pianoroll x, the output of pianoroll generation stage, and the pianorolls extracted from the final MIDI-event sequence. Illustrations of other fine-grained editing tasks and the final musical pieces can be found in our demo page https://sdmuse.github.io/posts/sdmuse/.

D. Method Analyses

1) Refinement Performance: Here we illustrate the refinement performance of the MIDI-event generation stage. We compare the pianorolls input of this stage and the pianorolls extracted from the output MIDI-event sequences as in Fig. 6. We manually added outliers to the input pianoroll to explore the ability of SDMuse in eliminating outliers. As shown in Fig. 6, these added outliers are removed after the MIDI-event generation stage. Also, the positions of some notes are refined in the MIDI-event sequence generation stage. For a more intuitive listening experience, please refer to our demo page.
page, where we post the audio synthesized from the output of two stages respectively for comparison.

2) Embedding Ways of Control Signals: When involving control signals \( c_e \) and \( c_p \) in the diffusion probabilistic model of the pianoroll generation stage, there are several embedding ways: 1) positional embedding: convert the control signals into sinusoidal positional embeddings; 2) direct embedding: regard control signals as vectors and use them as embeddings directly; 3) word embedding: convert the control signals to randomly initialized word embeddings. As shown in Fig. 5, the ways of direct embedding and word embedding show good performance in terms of the control signals’ faithfulness.

V. CONCLUSION

In this article, we propose SDMuse, a unified Stochastic Differential Music editing and generation framework via hybrid representations. SDMuse can not only compose whole musical pieces from scratch (both unconditionally and conditioned on given control signals), but also edit existing musical pieces in different ways according to various demands. As two different symbolic music representations, pianoroll is more appropriate for extracting and editing perceptive music information, such as structure, while MIDI-event is more ideal for generating and modeling music performance details. Thus, SDMuse contains pianoroll and MIDI-event generation stages to take advantage of hybrid representations. The first stage is based on a diffusion model generative prior and synthesizes or edits pianorolls by iteratively denoising through SDE. And the second stage refines pianorolls with music performance details by generating MIDI-event sequences auto-regressively. Objective and subjective results of a dataset demonstrate the effectiveness of our proposed stochastic differential music editing/generation process and hybrid representations. In the future, we plan to deploy SDMuse as an interactive website to make it accessible to more people who are interested in it, as well as extend it to other music genres.

ACKNOWLEDGMENT

This work was done during an internship at Sea AI Lab, Singapore.

REFERENCES

[1] B. D. Anderson, “Reverse-time diffusion equation models,” Stochastic Processes Their Appl., vol. 12, no. 3, pp. 313–326, 1982.
[2] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, “Symbolic music genre transfer with cycleGAN,” in Proc. IEEE 30th Int. Conf. Tools Artif. Intell., 2018, pp. 786–793.
[3] C.-J. Chang, C.-Y. Lee, and Y.-H. Yang, “Variable-length music score infilling via XLNet and musically specialized positional encoding,” in Proc. Int. Soc. Music Inf. Retrieval, 2021, pp. 97–104.
[4] Ö. Çöçek et al., “3D U-Net: Learning dense volumetric segmentation from sparse annotation,” in Proc. Int. Conf. Med. Image Comput. Comput.- Assisted Interv., 2016, pp. 424–432.
[5] O. Cifka, U. Şimşekli, and G. Richard, “Groove2Groove: One-shot music style transfer with style separation from synthetic data,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 28, pp. 2638–2650, 2020.
[6] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in Proc. AAAI Conf. Artif. Intell., 2018, vol. 32, pp. 34–41.
[7] F. Guo et al., “Automatic song translation for tonal languages,” in Proc. Findings Assoc. Comput. Linguistics, 2022, pp. 729–743.
[8] G. Hadjeres and L. Crestel, “The piano inpainting application,” 2021, arXiv:2107.05944.
[9] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in Proc. Adv. Neural Inf. Process. Syst., 2020, vol. 33, pp. 6840–6851.
[10] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, pp. 178–186.
[11] Y.-S. Huang and Y.-H. Yang, “Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 1180–1188.
[12] Z. Ju et al., “Telemelody: Lyric-to-melody generation with a template-based two-stage method,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2021, pp. 5426–5437.
[13] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, “DiffSinger: Singing voice generation via shallow diffusion mechanism,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 11020–11028.
[14] C. Meng et al., “SDEdit: Guided image synthesis and editing with stochastic differential equations,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–33.
[15] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, “Symbolic music generation with diffusion models,” in Proc. Int. Soc. Music Inf. Retrieval, 2021, pp. 468–475.
[16] A. Nichol et al., “GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models,” in Proc. Proc. 39th Int. Conf. Mach. Learn., vol. 162, 2022, pp. 16784–16804.
[17] C. Payne, “MuseNet,” OpenAI, 2019. [Online]. Available: openai.com/blog/musenet
[18] Y. Ren et al., “PopMAG: Pop music accompaniment generation,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 1198–1206.
[19] 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. Int. Conf. Mach. Learn., 2018, pp. 4364–4373.
[20] Z. Sheng et al., “Songmass: Automatic song writing with pre-training and alignment constraint,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, pp. 13798–13805.
[21] Y. Song et al., “Score-based generative modeling through Stochastic differential equations,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–36.
[22] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, vol. 30, pp. 6000–6010.
[23] M. Wang, Y.-K. Lai, Y. Liang, R. R. Martin, and S.-M. Hu, “Bigpicture: Data-driven image extrapolation using graph matching,” ACM Trans. Graph., vol. 33, no. 6, 2014, pp. 173:1–173:13.
[24] Z. Wang et al., “Songdriver: Real-time music accompaniment generation without logical latency nor exposure bias,” in Proc. 30th ACM Int. Conf. Multimedia, 2022, pp. 1057–1067.
[25] S.-L. Wu and Y.-H. Yang, “Musenmorphose: Full-song and fine-grained music style transfer with one transformer VAE,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 31, 2021, pp. 1953–1967.
[26] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, “Midinet: A convolutional generative adversarial network for symbolic-domain music generation,” in Proc. Int. Soc. Music Inf. Retrieval’2022, 2017, pp. 324–331.
[27] R. A. Yeh et al., “Semantic image inpainting with deep generative models,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5485–5493.
[28] C. Zhang et al., “Relyme: Improving lyric-to-melody generation by incorporating lyric-melody relationships,” in Proc. 30th ACM Int. Conf. Multimedia, 2022, pp. 1047–1056.

Chen Zhang received the master’s degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2023, interested in AI music, speech, and multi-modal learning.

Yi Ren received the master’s degree in computer science from Zhejiang University, Hangzhou, China, in 2022, interested in audiovisual, speech, and music. He is currently a Research Scientist with ByteDance AI Lab.

Kejun Zhang received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2010. He is currently a Professor of computer science at Zhejiang University. His research interests include music, affective computing, and multimedia arts and design.

Shuisheng Yan (Fellow, IEEE) has research interests that include computer vision, machine learning, and multimedia analysis. He is currently a Fellow of Singapore’s Academy of Engineering, AAAI, ACM, and IAPR.