Symbolic Music Generation with Non-Differentiable Rule Guided Diffusion

Yujia Huang [1]   **Adishree Ghatare** [1]   **Yuanzhe Liu** [2]   **Ziniu Hu** [1]   **Qinsheng Zhang** [3]   **Chandramouli S Sastry** [4 5]
Siddharth Gururani [3]   **Sageev Oore** [4 5]   **Yisong Yue** [1]

Abstract

We study the problem of symbolic music generation (e.g., generating piano rolls), with a technical focus on non-differentiable rule guidance. Musical rules are often expressed in symbolic form on note characteristics, such as note density or chord progression, many of which are non-differentiable which pose a challenge when using them for guided diffusion. We propose Stochastic Control Guidance (SCG), a novel guidance method that only requires forward evaluation of rule functions that can work with pre-trained diffusion models in a plug-and-play way, thus achieving training-free guidance for non-differentiable rules for the first time. Additionally, we introduce a latent diffusion architecture for symbolic music generation with high time resolution, which can be composed with SCG in a plug-and-play fashion. Compared to standard strong baselines in symbolic music generation, this framework demonstrates marked advancements in music quality and rule-based controllability, outperforming current state-of-the-art generators in a variety of settings. For detailed demonstrations, code and model checkpoints, please visit our project website.

1. Introduction

We are interested in developing methods for controllable symbolic music generation. There has been rapid progress in the development of modern generative models for symbolic music (Huang et al., 2018; Huang & Yang, 2020; Hsiao et al., 2021; Min et al., 2023). To facilitate interaction between human composers and these models, it is crucial for these models to adhere to specific musical rules, such as chord progression, during the composition process.

[1]California Institute of Technology [2]Rensselaer Polytechnic Institute [3]NVIDIA [4]Dalhousie University [5]Vector Institute. Correspondence to: Yujia Huang <yjhuang@caltech.edu>.
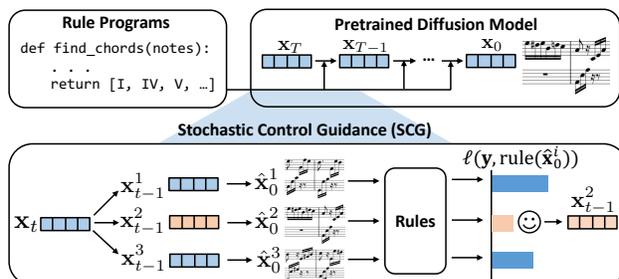
Figure 1. Overview of Stochastic Control Guidance (SCG) for plug-and-play non-differentiable rule guided generation. At each sampling step, we sample several realizations of the next step, and select the one yielding the most rule-compliant clean sample.

A common method to incorporate rules in generative models is to train with rule labels (Choi et al., 2020; Wu & Yang, 2023; von Rütte et al., 2022). However, integrating multiple musical rules during the training phase poses a significant challenge. Continuously updating model parameters to accommodate each new rule is not only costly but also will soon become impractical for compositions that involve many rules. Hence, there's a growing need for a method to guide pre-trained generative models in generating samples that conform to specific rules in a more flexible, light-weight, or plug-and-play manner.

Diffusion models (Ho et al., 2020; Song et al., 2021b) have emerged as a powerful generative modeling approach in many domains including images (Dhariwal & Nichol, 2021), audio (Huang et al., 2023) and video (Ho et al., 2022). A key feature of diffusion models is that they allow for post-hoc guidance of pre-trained models. Recent works have demonstrated success in guiding diffusion models with differentiable losses in a plug-and-play manner (Chung et al., 2023; Song et al., 2023). Starting from Gaussian noise, diffusion models generate samples from coarse to fine. The key idea of guidance is to update each intermediate step with the gradient of the loss. However, there are still two challenges to generate symbolic music with rule guidance: First, many rules (e.g. note density) are not differentiable. Second, they may be black box APIs that hinder backpropagation.

To this end, we propose Stochastic Control Guidance (SCG), a new algorithm that enables plug-and-play guidance in diffusion models for non-differentiable rules. Our algorithm

1

is inspired by stochastic control, where we pose the problem of generating samples that follow rule guidance as optimal control within a stochastic dynamical system. We obtain the analytical form of optimal control via path integral control theory (Theodorou et al., 2010), and adapt it to an efficient implementation within diffusion models. Specifically, we generate multiple realizations at each sampling step, and select the one that best follows the target (Figure 1). This process only requires forward evaluation of rule functions, making it applicable to non-differentiable rules.

To develop a practical overall framework, we also introduce a latent diffusion architecture with a transformer backbone for symbolic music generation. This architecture is able to generate dynamic music performances at 10ms time resolution, which is a significant challenge for standard pixel space diffusion models.

Our framework demonstrates state-of-the-art performance in various music generation tasks, offering superior rule guidance over popular methods and enabling musicians to effectively use it as a compositional tool. Our code is available here.

In summary, our contributions are as follows:

- We introduce Stochastic Control Guidance (SCG), which achieves plug-and-play guidance in diffusion models for non-differentiable rules.

- We provide a theoretical justification of SCG from a stochastic control perspective.

- We introduce a latent diffusion model architecture for symbolic music generation with high time resolution.

- We demonstrate that our framework enables flexible, interpretable and controllable symbolic music generation in a variety of tasks.

2. Related Works

Current symbolic music generation methods are mainly divided into MIDI token-based and piano roll-based approaches. MIDI-based methods treat music as sequences of discrete tokens, often using transformers for MIDI token generation (Huang et al., 2018; Huang & Yang, 2020; Ren et al., 2020; Hsiao et al., 2021). Piano roll representations, resembling image formats with time on the horizontal axis and pitches vertically, have inspired the use of image generative models like GANs (Yang et al., 2017; Dong et al., 2018) for their generation. Recent efforts (Atassi, 2023; Min et al., 2023) apply diffusion models to generate binary, quantized piano rolls. Our work extends this by incorporating velocity and pedal information into piano rolls and employing a finer time resolution of 10 ms, thereby facilitating the generation of more dynamic piano performances.

Another line of research seeks to enhance control over certain attributes in the generated music. Some studies (Brunner et al., 2018; Roberts et al., 2018) have leveraged VAE models to learn a disentangled latent space, achieving controllability over specific attributes by manipulating latents in designated directions. Further, various works have conditioned LSTMs (Meade et al., 2019) or transformers on different factors like style (Choi et al., 2020), note density (Wu & Yang, 2023), or attributes like time signature, instruments, and chords (von Rütte et al., 2022). However, these methods are limited to predefined attributes and are not easily extendable to new attributes due to the necessity of conditioning on labels during training.

Recent developments in the use of diffusion models for symbolic music generation have adapted controllable image generation techniques. Examples include generating complementary parts given melody/accompaniment (inpainting), bridging two music segments (infilling) (Min et al., 2023), extending existing music pieces (outpainting), and generating piano rolls from stroke piano rolls (Zhang et al., 2023a). Yet, when it comes to rule-based guidance, existing approaches still require training on specific attributes, such as chord progression (Min et al., 2023; Li & Sung, 2023), limiting their adaptability for composers desiring to incorporate new rules. Our work enables flexible rule-based guidance via SCG. Additionally, our method is compatible with other diffusion model techniques like inpainting, outpainting, and editing, further enhancing its versatility in music generation.

The conceptualization of stochastic optimal control in diffusion models has spurred theoretical advancements and practical applications. Path integral theory (Kappen, 2005) provides an efficient way of solving stochastic optimal control problems. Zhang & Chen (2021) employed it to transform a simple Ornstein–Uhlenbeck process to a novel process whose target distribution matches given marginal distribution. Further extending this framework, Berner et al. (2022); Vargas et al. (2023) established a novel link between stochastic optimal control problems and generative models, interconnected through stochastic differential equations.

3. Background

Score-based diffusion models. Diffusion models generate data by reversing a diffusion process. Let $p(\mathbf{x})$ be the unknown data distribution, the forward diffusion process $\{\mathbf{x}_t\}_{t \in [0,T]}$ diffuse $p(\mathbf{x})$ to a noise distribution that is easy to sample from (e.g. standard Gaussian distribution). Song et al. (2021b) models the forward diffusion process as the solution to an SDE:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \tag{1}$$

where the initial condition $\mathbf{x}_0 := \mathbf{x} \sim p(\mathbf{x})$, $\mathbf{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is the drift coefficient, $g : \mathbb{R} \to \mathbb{R}$ is the diffusion

coefficient and $\mathbf{w} \in \mathbb{R}^d$ is a standard Wiener process.

Let $p_t(\mathbf{x})$ denote the marginal distribution of $\mathbf{x}_t$. The diffusion and drift coefficient can be properly designed such that $p_T(\mathbf{x}) \approx \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. In this paper, we consider the VP-SDE (Song et al., 2021b), where $\mathbf{f}(\mathbf{x}, t) := -\frac{1}{2}\beta(t)\mathbf{x}$ and $g(t) := \sqrt{\beta(t)}$, where $\beta(t)$ is a noise schedule. DDPM (Ho et al., 2020) can be regarded as a discretization of VP-SDE.

Samples are generated using the reverse-time SDE:

$$d\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] dt + g(t) d\bar{\mathbf{w}}_t, \quad (2)$$

where $\mathbf{f}(\mathbf{x}_t, t) : \mathbb{R}^d \to \mathbb{R}$ is the drift coefficient, $g : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient, dt is an infinitesimal negative time step and $\bar{\mathbf{w}}_t$ is a standard reverse-time Wiener process. Sampling $\mathbf{x}_T \sim p_T(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solving the above SDE from $t = T$ to $t = 0$ produces samples from the data distribution: $\mathbf{x}_0 \sim p_0(\mathbf{x}) = p(\mathbf{x})$.

Since the data distribution is unknown, it is popular to approximate the score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ via a neural network $\mathbf{s}_\theta(\mathbf{x}, t)$ and train it with a weighted sum of denoising score matching objectives (Song et al., 2021b).

Classifier and Classifier-free Guidance. Guided diffusion models generates samples from $p(\mathbf{x}|\mathbf{y})$ given label $\mathbf{y}$. Classifier guidance (Dhariwal & Nichol, 2021) achieves this by training a classifier $p_t(\mathbf{y}|\mathbf{x}_t)$ on the noisy sample and label pair, and mix its gradient with the score of the diffusion model during sampling. The conditional score function becomes $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \omega \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t)$, where ω is called guidance scale. This approximates the samples from the distribution $\tilde{p}(\mathbf{x}_t|\mathbf{y}) \propto p(\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_t)^\omega$. Classifier guidance is able to guide a pre-trained generative model at the cost of training an extra classifier on the noisy data.

Classifier-free guidance (Ho & Salimans, 2022) avoids training classifiers by jointly training conditional and unconditional diffusion models, and combining their score estimates during sampling. The mixed score function becomes $(1+\omega)\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y}) - \omega \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, where ω is the guidance strength. Despite easy implementation, it is expensive to extend classifier-free guidance to unknown or composite labels, because it requires re-training the diffusion model.

Loss-Guided Diffusion. To reduce the need of additional training for conditional generation, methods have been proposed to guided diffusion models to generate samples in a plug-and-play way. Instead of training a classifier to approximate $p(\mathbf{y}|\mathbf{x}_t)$, Diffusion Posterior Sampling (DPS) (Chung et al., 2023) uses $p(\mathbf{y}|\hat{\mathbf{x}}_0)$, where $\hat{\mathbf{x}}_0 := \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ is obtained through the Tweedie's formula (Efron, 2011):

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}(t)}}(\mathbf{x}_t + (1 - \bar{\alpha}(t))\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)). \quad (3)$$

Recall that $p(\mathbf{y}|\mathbf{x}_t)$ can be factorized as:

$$p(\mathbf{y}|\mathbf{x}_t) = \int p(\mathbf{y}|\mathbf{x}_0)p(\mathbf{x}_0|\mathbf{x}_t)d\mathbf{x}_0 = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}p(\mathbf{y}|\mathbf{x}_0).$$

DPS uses a point estimation of this quantity. Later work (Song et al., 2023) proposes to use Monte-Carlo estimation of this by sampling from approximated $p(\mathbf{x}_0|\mathbf{x}_t)$. However, these methods requires the loss function used to specify the condition to be differentiable. Many symbolic rules we consider in this paper are non-differentiable.

4. Non-Differentiable Rule Guidance

We now present Stochastic Control Guidance for non-differentiable rule guidance in diffusion models. We start with defining rule guidance in Section 4.1. Inspired by stochastic control (Section 4.2), we define a value function as a loss measuring (lack of) rule adherence, and show that optimal control steers the reverse diffusion to the target distribution. We then discuss practical algorithms (Section 4.3). We conclude by establishing a general theoretical connection that enables many guidance methods to be viewed through the lens of stochastic optimal control (Section 4.4).

4.1. Rule Guidance Problem

Assume that we have a pre-trained diffusion model that can sample from the data distribution $p(\mathbf{x})$, and a loss function $\ell_\mathbf{y} : \mathcal{X} \to \mathbb{R}$ that characterizes how well a sample follows some conditions $\mathbf{y}$: $p(\mathbf{y}|\mathbf{x}) \propto e^{-\ell_\mathbf{y}(\mathbf{x})}$. Our goal is to sample from the following distribution:

$$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x})\frac{e^{-\ell_\mathbf{y}(\mathbf{x})}}{Z} \propto p(\mathbf{x})p(\mathbf{y}|\mathbf{x}), \quad (4)$$

where $Z = \int_\mathbf{x} p(\mathbf{x})e^{-\ell_\mathbf{y}(\mathbf{x})}d\mathbf{x}$.

A central challenge that we tackle is that many musical rules are non-differentiable, which makes sampling from Eq. 4 difficult. For instance, let $\mathbf{x} = [x_1, x_2, ..., x_n] \in [0, 1]^n$ be a vector where each x_i represents the volume of a note, so that the note density is computed as $\text{ND}(\mathbf{x}) = \sum_{i=1}^n \mathbb{1}(x_i > \epsilon)$, where ϵ is a small number. Let $\mathbf{y} = y$ be a scalar that represents the target note density. Then the loss is defined as $\ell_\mathbf{y}(\mathbf{x}) = |y - \text{ND}(\mathbf{x})|$, which is non-differentiable.

4.2. Guidance via Stochastic Control

The pre-trained diffusion model generates samples using the reverse-time SDE (Eq. 2). Let $\boldsymbol{\eta}_t = \mathbf{x}_{T-t}$, and $\tilde{\mathbf{f}}(\boldsymbol{\eta}_t, t) = \mathbf{f}(\boldsymbol{\eta}_t, t) - g(t)^2 \nabla_{\boldsymbol{\eta}_t} \log p_t(\boldsymbol{\eta}_t)$. We can rewrite Eq. 2 as:

$$d\boldsymbol{\eta}_t = \tilde{\mathbf{f}}(\boldsymbol{\eta}_t, t)dt + g(t)d\mathbf{w}_t, \quad (5)$$

where dt is an infinitesimal time step and $d\mathbf{w}$ is a standard Wiener process. Sampling $\boldsymbol{\eta}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solving the

above SDE from $t = 0$ to $t = T$ produces samples from the data distribution.

We want to find a control $\mathbf{u}(\boldsymbol{\eta}_t, t)$, such that solving the following SDE yields samples from target distribution $p(\boldsymbol{\eta}|\mathbf{y})$:

$$\mathrm{d}\boldsymbol{\eta}_t = \tilde{\mathbf{f}}(\boldsymbol{\eta}_t, t)\mathrm{d}t + g(t)(\mathbf{u}(\boldsymbol{\eta}_t, t)\mathrm{d}t + \mathrm{d}\mathbf{w}_t). \quad (6)$$

We use $\mathbf{u}_t := \mathbf{u}(\boldsymbol{\eta}_t, t)$ and $\tilde{\mathbf{f}}_t := \tilde{\mathbf{f}}(\boldsymbol{\eta}_t, t)$ for brevity, noting they are state-dependent.

Considering the stochastic dynamical system in Eq. 6 for $0 \le t \le T$ and initial state $\boldsymbol{\eta}_0 = \bar{\boldsymbol{\eta}}_0$, we address the optimal control problem associated with the cost function $C_u(\boldsymbol{\eta}_t, t)$, which is defined as the expectation over all stochastic trajectories starting at $\boldsymbol{\eta}_t$ with control function $\mathbf{u}_t$:

$$C_u(\boldsymbol{\eta}_t, t) = \mathbb{E}\left[\phi(\boldsymbol{\eta}_T) + \int_t^T \frac{1}{2}\|\mathbf{u}_t\|^2 \mathrm{d}t\right]. \quad (7)$$

It is known that the optimal control policy admits an analytical solution (Pavon, 1989):

$$\mathbf{u}_t^* = -g(t)\nabla_{\boldsymbol{\eta}}V(\boldsymbol{\eta}, t), \quad (8)$$

where function $V(\boldsymbol{\eta}, t)$, known as the *value* function, is the solution to celebrated stochastic Hamilton-Jacobi-Bellman (HJB) equation (Evans, 2022):

$$-\partial_t V(\boldsymbol{\eta}, t) = -\frac{1}{2}g(t)^2(\nabla_{\boldsymbol{\eta}}V)^\top(\nabla_{\boldsymbol{\eta}}V)$$
$$+ (\nabla_{\boldsymbol{\eta}}V)^\top \tilde{\mathbf{f}}_t + \frac{1}{2}g(t)^2\mathrm{Tr}(\nabla_{\boldsymbol{\eta}\boldsymbol{\eta}}^2 V), \quad (9)$$

with boundary condition $V(\boldsymbol{\eta}, T) = \phi(\boldsymbol{\eta})$.

Path Integral Control. Although solving HJB in Eq. 9 is nontrivial due to its non-linearity w.r.t. V, using an exponential transformation $\Psi(\boldsymbol{\eta}, t) = e^{-V(\boldsymbol{\eta}, t)}$ yields a linear HJB equation in Ψ:

$$-\partial_t \Psi(\boldsymbol{\eta}, t) = \left(\tilde{\mathbf{f}}_t^\top \nabla_{\boldsymbol{\eta}} + \frac{1}{2}g(t)^2\mathrm{Tr}(\nabla_{\boldsymbol{\eta}\boldsymbol{\eta}}^2)\right)\Psi(\boldsymbol{\eta}, t),$$
$$(10)$$

with boundary condition $\Psi(\boldsymbol{\eta}, T) = e^{-\phi(\boldsymbol{\eta})}$. We call Ψ the *desirability* function as it is inversely related to the value V.

Let $\Omega = C([0, T]; \mathbb{R}^d)$ be the space consisting of all possible continuous-time stochastic trajectories $\tau = \{\boldsymbol{\eta}_t, 0 \le t \le T\}$, and $\mathcal{Q}^0$ be the measure induced by an uncontrolled stochastic process (Eq. 5). Then the linear HJB equation has the following solution according to the Feynman-Kac formula (Øksendal, 2003):

$$\Psi(\boldsymbol{\eta}, t) = \mathbb{E}_{\mathcal{Q}^0}\left[e^{-\phi(\boldsymbol{\eta}_T)}|\boldsymbol{\eta}_t = \boldsymbol{\eta}\right]. \quad (11)$$

Eq. 11 shows that the value function can be computed by *only forward sampling the uncontrolled process* without

---

Algorithm 1 Stochastic Control Guided DDPM sampling

Require: Loss function ℓ_y, rule target $\mathbf{y}$, number of samples n, forward process variances β_t, $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$.
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ **to** 1 **do**
  ▷ Compute the posterior mean of $\mathbf{x}_{t-1}$.
  $\hat{\mathbf{x}}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$
  if $t > 1$ **then**
    ▷ Sampling possible next steps.
    $\mathbf{x}_{t-1}^i = \hat{\mathbf{x}}_{t-1} + \sigma_t \mathbf{z}^i$, with $\mathbf{z}^1, ..., \mathbf{z}^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    ▷ Estimate the clean sample from noisy sample.
    $\hat{\mathbf{x}}_0^i = \frac{1}{\sqrt{\bar{\alpha}_{t-1}}}\left(\mathbf{x}_{t-1}^i - \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(\mathbf{x}_{t-1}^i, t-1)\right)$
    ▷ Find the direction that minimizes the loss.
    $k = \arg\max_i \log p(y|\hat{\mathbf{x}}_0^i) = \arg\max_i -\ell_y(\hat{\mathbf{x}}_0^i)$
    $\mathbf{x}_{t-1} = \mathbf{x}_{t-1}^k$
  else
    $\mathbf{x}_{t-1} = \hat{\mathbf{x}}_{t-1}$
  end if
end for
return: $\mathbf{x}_0$

---

knowing the optimal control policy. Plugging Eq. 11 into Eq. 8 yields the analytic optimal policy, which aligns with the well-known path integral control approach (Theodorou et al., 2010; Theodorou, 2015; Fleming & Mitter, 1982):

$$\mathbf{u}_t^*(\boldsymbol{\eta})\mathrm{d}t = g(t)\nabla_{\boldsymbol{\eta}}\log\Psi(\boldsymbol{\eta}, t)\mathrm{d}t \quad (12)$$
$$= \frac{\mathbb{E}_{\mathcal{Q}^0}\left[e^{-\phi(\boldsymbol{\eta}_T)}\mathrm{d}\mathbf{w}_t|\boldsymbol{\eta}_t = \boldsymbol{\eta}\right]}{\mathbb{E}_{\mathcal{Q}^0}\left[e^{-\phi(\boldsymbol{\eta}_T)}|\boldsymbol{\eta}_t = \boldsymbol{\eta}\right]}. \quad (13)$$

Next, we show that using the above optimal control, we can guide the generation process to produce samples from the target conditional distribution $p(\boldsymbol{\eta}|\mathbf{y})$.

Theorem 4.1 (proof in Appendix A.1). *Consider the dynamical system in Eq. 6. For a terminal cost defined as $\phi(\boldsymbol{\eta}_T) \triangleq \ell_y(\boldsymbol{\eta}_T) \triangleq -\log p(\mathbf{y}|\boldsymbol{\eta}_T) + \mathtt{const}$, and initial condition $\boldsymbol{\eta}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the terminal distribution induced by the optimal control policy $\mathbf{u}_t^*$ (Eq. 13) is:*

$$\mathcal{Q}^*(\boldsymbol{\eta}_T) = p(\boldsymbol{\eta}_T|\mathbf{y}). \quad (14)$$

4.3. Practical Algorithms

Approximation of the Optimal Control. In practice, it is expensive to compute Eq. 13, because one needs to unroll the whole trajectory to get $\boldsymbol{\eta}_T$. Instead of using Eq. 13 as our optimal control, we set $\mathbf{u}_t\mathrm{d}t + \mathrm{d}\mathbf{w}$ to the following:

$$\underset{\mathrm{d}\mathbf{w}_t}{\arg\max} -\ell_y(\hat{\boldsymbol{\eta}}_T), \quad (15)$$

where $\hat{\boldsymbol{\eta}}_T = \mathbb{E}[\boldsymbol{\eta}_T|\boldsymbol{\eta}_{t+\mathrm{d}t}]$ can be obtained via Tweedie's Formula (Eq. 3), which is a one-step computation and much cheaper than solving the whole trajectory.

Eq. 15 is an approximation to a tempered version of Eq. 13. Consider the terminal cost is defined with a scaling factor K, i.e. $\phi(\boldsymbol{\eta}_T) = \ell_y(\boldsymbol{\eta}_T)/K$. When $K \to 0$, Eq. 13 becomes:

$$\operatorname*{argmax}_{\mathrm{d}\mathbf{w}_t} \max_{\tau} -\ell_y(\boldsymbol{\eta}_T|\boldsymbol{\eta}_{t+\mathrm{d}t}), \qquad (16)$$

where $\boldsymbol{\eta}_{t+\mathrm{d}t} = \boldsymbol{\eta}_t + \tilde{\mathbf{f}}(\boldsymbol{\eta}_t, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t$, and $\tau : [t + \mathrm{d}t, T] \to \mathbb{R}^d$ represents a trajectory. The solution of Eq. 15 optimizes a lower bound of the objective in Eq. 16:

$$\max_{\mathrm{d}\mathbf{w}_t, \tau} -\ell_y(\boldsymbol{\eta}_T|\boldsymbol{\eta}_{t+\mathrm{d}t}) \geq \max_{\mathrm{d}\mathbf{w}_t} -\ell_y(\mathbb{E}\left[\boldsymbol{\eta}_T|\boldsymbol{\eta}_{t+\mathrm{d}t}\right]). \quad (17)$$

Intuition. Our SCG algorithm implemented with DDPM sampling (Ho et al., 2020) is outlined in Algorithm 1 and illustrated in Figure 1, where we use $\mathbf{x}_t \triangleq \boldsymbol{\eta}_{T-t}$ to denote the intermediate states following conventions of diffusion model notations. The intuition is that we select the direction that leads to the most probable sample at each step. For every step t in the sampling process, given $\mathbf{x}_t$, we compute multiple realizations of the next step $\mathbf{x}_{t-1}$, estimate the corresponding clean sample $\hat{\mathbf{x}}_0$, and choose the $\mathbf{x}_{t-1}$ that leads to the lowest loss $\ell_{\mathbf{y}}(\hat{\mathbf{x}}_0)$. Notably, we only need to evaluate the forward pass of the rule function, and there is no need to evaluate or estimate its gradient, making our method suitable for non-differentiable and black-box rule functions. Furthermore, it is also compatible with other stochastic sampling procedure in diffusion models (Appendix B).

4.4. General Theoretical Connection

In this section, we show a general connection (Proposition 4.2) that enables many guidance methods to be viewed through the lens of stochastic optimal control.

Proposition 4.2 (proof in Appendix A.2). *Consider the dynamical system in Eq. 6 with terminal cost* $\phi(\boldsymbol{\eta}_T) \triangleq -\log p(\mathbf{y}|\boldsymbol{\eta}_T) + \mathtt{const}$. *We have:* $\Psi(\boldsymbol{\eta}_t, t) = c \cdot p(\mathbf{y}|\boldsymbol{\eta}_t)$.

Proposition 4.2 says that the desirability function equals to the likelihood function. Then many popular guidance techniques can be seen as different implementations of the optimal control following Eq. 12):

$$g(t)\nabla_{\boldsymbol{\eta}_t} \log p(\mathbf{y}|\boldsymbol{\eta}_t) = g(t)\nabla_{\boldsymbol{\eta}_t} \log \Psi(\boldsymbol{\eta}_t, t) = \mathbf{u}_t^*(\boldsymbol{\eta}_t).$$

Classifier guidance (Dhariwal & Nichol, 2021) trains a neural network on noisy data pair $\{\boldsymbol{\eta}_t, \mathbf{y}\}$ to approximate $\Psi(\boldsymbol{\eta}_t, t)$, and differentiate through it to obtain $\mathbf{u}_t^*(\boldsymbol{\eta})$.

DPS (Chung et al., 2023) avoids training a surrogate model by approximating $\Psi(\boldsymbol{\eta}_t, t)$ with $\Psi(\hat{\boldsymbol{\eta}}_T, T)$, where $\hat{\boldsymbol{\eta}}_T$ is the posterior mean that can be obtained through the Tweedie's formula (Eq. 3). Since $\nabla_{\boldsymbol{\eta}_t}\Psi(\hat{\boldsymbol{\eta}}_T, T) = \frac{\partial \Psi(\hat{\boldsymbol{\eta}}_T, T)}{\partial \hat{\boldsymbol{\eta}}_T}\frac{\partial \hat{\boldsymbol{\eta}}_T}{\partial \boldsymbol{\eta}_t}$, it requires $\Psi(\hat{\boldsymbol{\eta}}_T, T) \propto e^{-\ell_{\mathbf{y}}(\hat{\boldsymbol{\eta}}_T)}$ to be differentiable.

In contrast, our approach is inspired by path integral control, and only needs the forward evaluation of the rule function (Eq. 15). Therefore, our method does not require the rule function to be differentiable.

5. Latent Diffusion Architecture

To arrive at a practical overall framework, we develop a latent diffusion architecture tailored towards symbolic music generation, and in particular able to generate at 10ms time resolution. This architecture can be combined with Stochastic Control Guidance in a plug-and-play fashion.

Data Representation. We represent symbolic music as a 3-channel tensor. Each column in this representation accounts for a 10 ms timeframe. The first channel is the piano roll, where horizontal axis represents time and vertical axis represents pitch. Each element takes value from 0-127, indicating the velocity (volume) of the note. The second channel is the onset roll, consisting of binary values that denote the presence of note onsets. The third channel is the pedal roll, representing the sustain pedal control for each timeframe.

Model architecture. We first use a VAE model to encode short segments of piano rolls of shape $3 \times 128 \times 128$ into a latent space. Then we concatenate the latent codes and train a diffusion model to capture their joint distribution (Figure 2). For the VAE, we use the same architecture following (Rombach et al., 2022). The training involves a denoising objective in conjunction with KL regularization: we introduce musically semantic perturbations (such as adding adjacent notes) to the data and train the model to revert to the original, unperturbed data. Both KL regularization and the denoising objective have proven indispensable for developing diffusion models with robust generative capabilities in subsequent stages.
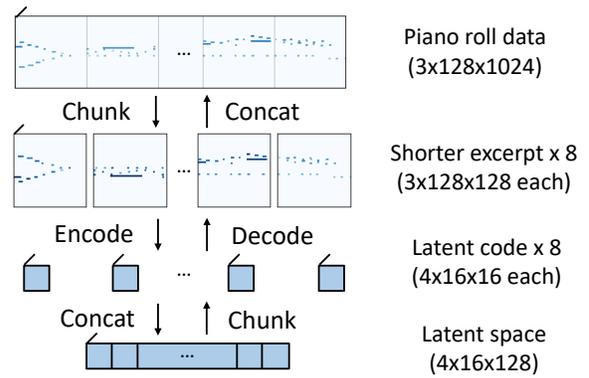

Figure 2. We use a VAE to encode piano roll segments to latent space and concatenate them for the next stage of diffusion training.

For the diffusion model, we use the DiT architecture (Peebles & Xie, 2023). In contrast to the standard U-Net, the

transformer backbone is more adept at handling sequences of latent tokens. Rather than absolute position encoding, we use rotary position embedding (Su et al., 2023) to better generalize across various input lengths. We train the diffusion model on piano rolls of length 1024 (10.24 s). To generate musical excerpts of arbitrary length, we apply DiffCollage (Zhang et al., 2023b) to aggregate the score function of shorter music segments.

6. Experiments

We evaluate our method on a wide range of symbolic music generation tasks: unconditional generation (Sec 6.2), individual rule guidance (Sec 6.3), composite rule guidance (Sec 6.4) and editing (Appendix C.2). We perform ablation studies in Sec 6.5 and subjective evaluation in Sec 6.6. In addition, we demonstrate that our method can be used as a compositional tool for musicians in Sec 6.7.

6.1. Experimental Settings

Data. We train our model on several piano midi datasets that cover both classical and pop genres. The MAESTRO dataset (Hawthorne et al., 2019) has about 1200 pieces of classical piano performances with expressive dynamics, resulting in about 200 hours of music performance. In addition, we crawled about 14k MIDI files from Muscore in classical, religion, and soundtrack genres across all skill levels, yielding about 700 hours of data. We also used two Pop piano datasets: Pop1k7 (Hsiao et al., 2021) and Pop909 (Wang et al., 2020) that contain 108 hours and 60 hours of pop piano midi files.

Training and Inference Setup. We first train a VAE model to encode piano rolls to latent space, then fix the VAE and train a diffusion model on this space. The diffusion model is trained with dataset-based conditioning: classical performance (Maestro), classical sheet music (Muscore) and Pop (pop1k7 and pop909), following Classifier-free Guidance (Ho & Salimans, 2022) with a dropout rate of 0.1. We train the model for 1.2M steps and use DDPM (Ho et al., 2020) with 1000 steps as the default sampling method unless stated otherwise. All experiments are run on NVIDIA A100-SXM4 GPUs.

6.2. Unconditional Generation

Baselines. We compare with state-of-the-art symbolic music generators trained on various datasets (Table 1).

Method	Model	Dataset	Representation
MusicTr (Huang et al., 2018)	Transformer	Maestro	MIDI-like
Remi (Huang & Yang, 2020)	Transformer	Pop775	REMI
CPW (Hsiao et al., 2021)	Transformer	Pop1k7	CP
PolyDiff (Min et al., 2023)	Diffusion	POP909	Piano roll

Table 1. Baselines for unconditional music generation.

Objective Metrics. It is worth mentioning that quantitative evaluation of music quality remains an open problem (Yin et al., 2023). Nevertheless, we use the average overlapping area (OA) between the intra-set and inter-set distribution of 7 musical attributes (pitch range, note density, etc.) proposed in (Yang & Lerch, 2020) as the objective metric for music quality. As a sanity check, we compare a subset of the training dataset with another subset (denoted by GT in Table 2), and find that GT on all the datasets achieves the highest average OA. This indicates that this metric is a reasonable necessary condition for good generated music quality.

Results. The evaluation results are in Table 2, highlighting the highest values (excluding GT) in bold. Our method achieves the highest average OA on all the datasets. The detailed results for all 7 OA metrics are in Table 7, Appendix C. The baselines are trained on individual dataset, and do not generalize well across datasets. MusicTr has the second-best overall rating for classical music (Maestro and Muscore), while it holds the lowest rating for pop music. CPW, on the other hand, ranks second in pop music but has the lowest rating in classical music. In contrast, our model delivers strong performance consistently across all the datasets.

6.3. Individual Rule Guidance

Setup. We consider three rules: pitch histogram, note density (vertical and horizontal) and chord progression, where pitch histogram is differentiable and the other two are non-differentiable (see Appendix D.1 for the full definition of each rule). In our evaluation of the guidance performance, we default to conditioning on the Muscore dataset unless otherwise specified, owing to its comprehensive variety and extensive coverage of a broad spectrum of rule labels. For each rule, we randomly select 200 samples from the test dataset, and extract their attributes as the target for guided generation. We choose the number of samples to be 16 for SCG if without explicit mentioning.

Baselines. We compare with two popular post-hoc guidance methods: classifier guidance (Dhariwal & Nichol, 2021) and Diffusion Posterior Sampling (DPS) (Chung et al., 2023). For classifier guidance, we train a classifier on noisy latent and target pair for each rule. DPS only requires the loss to be defined on clean data x_0 so we can directly plug in the rule in the loss if the rule is differentiable (DPS-Rule) without any additional training. However, it still requires the gradient of the rule, and therefore we train a surrogate model (a neural network) for non-differentiable rules (DPS-NN).

Metrics. We evaluate conditional generation performance by two metrics: loss and OA. Loss reflects controllability: whether the generated samples follow the target rules. For pitch histogram and note density, we use L2 loss, and for chord progression, we use 0-1 loss. However, a low loss does not necessarily indicates good quality music. For

Dataset	GT	MusicTr	Remi	CPW	PolyDiff	**SCG (ours)**
Maestro	0.944 ± 0.002	0.903 ± 0.005	0.847 ± 0.005	0.801 ± 0.006	0.842 ± 0.007	$\mathbf{0.943 \pm 0.003}$
Muscore	0.945 ± 0.004	0.901 ± 0.004	0.879 ± 0.006	0.843 ± 0.007	0.845 ± 0.004	$\mathbf{0.934 \pm 0.003}$
Pop	0.957 ± 0.002	0.845 ± 0.004	0.866 ± 0.004	0.899 ± 0.005	0.883 ± 0.004	$\mathbf{0.939 \pm 0.004}$

Table 2. Average Overlapping Area (OA) across seven music attributes for unconditional generation, with highest non-GT OA bolded.

Method	Loss $\downarrow$ (PH)	OA $\uparrow$ (PH)	Loss $\downarrow$ (ND)	OA $\uparrow$ (ND)	Loss $\downarrow$ (CP)	OA $\uparrow$ (CP)
No Guidance	0.018 ± 0.010	0.842 ± 0.012	2.486 ± 3.530	0.830 ± 0.016	0.831 ± 0.142	$\mathbf{0.854 \pm 0.026}$
Classifier	0.005 ± 0.004	$\underline{0.855 \pm 0.020}$	$\underline{0.698 \pm 0.587}$	$\mathbf{0.861 \pm 0.025}$	0.723 ± 0.200	0.850 ± 0.033
DPS - NN	$\mathbf{0.001 \pm 0.002}$	0.849 ± 0.018	1.261 ± 2.340	0.667 ± 0.113	$\underline{0.414 \pm 0.256}$	0.839 ± 0.039
DPS - Rule	0.010 ± 0.008	0.635 ± 0.006	2.508 ± 2.798	0.800 ± 0.080	-	-
SCG (ours)	$\underline{0.003 \pm 0.004}$	$\mathbf{0.867 \pm 0.005}$	$\mathbf{0.131 \pm 0.325}$	$\underline{0.842 \pm 0.031}$	$\mathbf{0.273 \pm 0.1637}$	$\underline{0.851 \pm 0.011}$

Table 3. Objective evaluation for individual rule guidance. Bottom 2 losses and top 2 OA metrics are highlighted. SCG significantly improves the controllability of non-differentiable rules.

instance, a music piece that follows the note density requirement may have random pitch and does not sound good. Therefore, we use the OA to measure how close the generated music distribution and the ground truth music distribution (with similar target attributes as the generated music) when projected to some music features space. Please see Appendix D.3 for the detailed evaluation setup.

Results. Table 3 shows the results. We make three main observations. First, our method significantly outperforms the other methods in generating samples that follow the *non-differentiable* rules (note density and chord progression). It achieves the lowest loss, without need for training any surrogate model, which is mandatory for classifier guidance and DPS-NN. However, it sacrifices OA a bit. This is because over-optimizing for one attribute will overlook the other attributes. Reducing the number of samples used for SCG can lead to better OA at the cost of a higher loss (See Section 6.5 on this trade-off).

Second, we find it challenging to train neural network surrogate models to approximate non-differentiable rules (Appendix E), leading to poor performance of guidance methods that rely on surrogate models. For differentiable rules (pitch histogram), the surrogate model learns well and DPS-NN achieves the lowest loss.

Third, to the best of our knowledge, our method is the first plug-and-play guidance method that supports non-differentiable and black-box loss functions. In contrast, DPS-rule fails to guide on note density because the gradient is zero almost everywhere. It also does not apply to chord progression because the loss involves a black-box API that cannot be back-propagated through. Overall, our method proves especially beneficial for guiding the generation process with non-differentiable loss functions, or for achieving guidance without the need for additional training.

6.4. Composite Rule Guidance

We apply our method to generate samples that follow composite rules, following the same setup in Section 6.3, and assuming that the rule labels are conditionally independent given the sample. For classifier and DPS-NN, we train a surrogate model for each rule, and combine the gradient of each classifier to obtain the guidance term: $\Sigma_i \omega_i \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}_i | \mathbf{x}_t)$. For our method, we use a weighted loss function $\Sigma_i \omega_i \ell_{\mathbf{y}_i}(\mathbf{x})$ to select the best direction for each step. We set the weights on pitch histogram, note density and chord progression to be 40, 1, 1 respectively so that their loss is on the same order of magnitude. In addition, we compared with four established conditional music generation baselines: Retrieval, Figaro-expert, Figaro-expert-learned (von Rütte et al., 2022) and MuseCOCO (Lu et al., 2023). The Retrieval baseline is: given a set of target attributes, we find the sample within the datasets that has the closest attributes to the target attributes. This serves as an oracle baseline.

Method	PH $\downarrow$	ND $\downarrow$	CP $\downarrow$	OA $\uparrow$
Retrieval	0.006 ± 0.005	0.433 ± 1.068	0.556 ± 0.182	$\mathbf{0.886 \pm 0.005}$
Figaro expert	0.007 ± 0.007	2.303 ± 2.256	0.761 ± 0.187	0.857 ± 0.047
Figaro expert+learned	0.006 ± 0.009	1.489 ± 2.737	0.726 ± 0.214	0.883 ± 0.008
MuseCoCo	0.040 ± 0.026	2.734 ± 3.551	0.821 ± 0.163	0.753 ± 0.038
No Guidance	0.018 ± 0.010	2.486 ± 3.530	0.831 ± 0.142	0.803 ± 0.096
Classifier	0.006 ± 0.006	0.822 ± 0.844	0.724 ± 0.205	0.859 ± 0.026
DPS-NN	0.004 ± 0.006	1.366 ± 2.265	0.661 ± 0.257	0.752 ± 0.079
SCG	0.014 ± 0.009	0.466 ± 0.648	0.446 ± 0.205	$\mathbf{0.874 \pm 0.007}$
DPS-NN + SCG	$\mathbf{0.002 \pm 0.007}$	0.238 ± 0.531	0.313 ± 0.231	0.781 ± 0.084
Classifier + SCG	0.003 ± 0.005	$\mathbf{0.148 \pm 0.203}$	$\mathbf{0.284 \pm 0.197}$	0.864 ± 0.010

Table 4. Objective evaluation for composite rule guidance. SCG + Classifier achieves significantly lower losses for all three rules simultaneously with a high OA score.

The results are presented in Table 4. We can see that, overall, our method achieves the lowest loss among all the baselines. The OA ranking is also higher than that of individual rule guidance, because guiding the generation with three attributes prevents over-optimizing for one attribute. There

are two baselines that have higher OA than us: retrieval and Figaro-expert-learned. It is as expected that retrieval has the highest OA because the samples are selected from the dataset rather than being generated. Figaro-expert-learned also achieves high OA, because it is a style transfer model that needs to take in source music rather than a generative model. The generated music is conditioned on the latent representation of the source music. Therefore, it is not a fair comparison but we still put it here for reference. MuseC-OCO cannot generate well following the target rules because it does not support fine-grained control.

Among the diffusion guidance methods, our method achieves much lower loss on non-differentiable rules compared to other methods, similar to the case of individual rule guidance. However, it compromises control over the pitch histogram. Additionally, the loss associated with each rule is higher than in scenarios of individual rule guidance, which aligns with expectations. This increase in loss occurs because it is more challenging to identify a direction that satisfies multiple rules simultaneously, as opposed to a single rule, within the same computational budget.

To enhance the controllability of composite rules, we integrate our SCG approach with gradient-based guidance methods. In this framework, the gradient of the surrogate model provides a preliminary guidance signal. SCG then identifies the optimal directions along these initially guided trajectories. As indicated in Table 4, this combination of our method with the baseline gradient method results in improved controllability for each rule, compared to the baseline method alone. Furthermore, we achieve a level of controllability comparable to that of individual rule guidance, while using the same number of samples.

6.5. Ablation Studies

Controllability and Computational Time Trade-off. Table 5 shows the loss achieved by SCG with different number of samples at each step. The time is reported for generating 4 samples in a batch. As anticipated, more samples results in lower loss, but requires more time. To achieve a balance between controllability and computational efficiency, we integrate classifier guidance with SCG. This combination yields interesting results: number of samples of 4, when used in conjunction with classifier guidance, delivers similar performance to number of samples of 16 with SCG alone, but is approximately four times faster.

Controllability and OA Trade-Off. We observe a trade-off between controllability (measured by loss) and OA (used as a necessary, albeit not sufficient, indicator of music quality.) in Table 5, where we guide the model to generate music following given note density. To better evaluate the trade-off, we compute 2 OA metrics, one uses the full dataset as reference (denoted by 'OA full'), the other uses the data

that comply with the desired rule (denoted by 'OA' as in previous sections). The motivation of 'OA full' stems from our approach of extracting note density from a diverse selection of sources within the dataset for conditional generation. If the generated pieces precisely mirror their source, they can achieve a low loss because they comply with the rule perfectly, and have a high OA because their sources are taken from the dataset (the 'source' row in Table 5).

We highlight the highest OA for each group of methods (the baselines, SCG and classifier+SCG with different number of samples). Note that the main difference between these two metrics is for 'No Guidance', where 'OA cluster' is much worse than 'OA full'. This is because the 'OA cluster' metric rewards controllability more. However the controllability and OA trade-off for SCG related methods are consistent among the two OA metrics. We think that this trade-off is caused by over-optimizing over some constraints. For instance, a generative model could generate music that follows the note density exactly, but completely ignore the pitch of the notes. One can balance controllability and OA by tuning the number of samples at each step.

Method	n	Loss ↓	OA full ↑	OA ↑	Time (s)
No Guidance	-	2.486 ± 3.530	0.918 ± 0.005	0.830 ± 0.016	25.4
Source	-	0	$\mathbf{0.923 \pm 0.008}$	-	-
Classifier	1	0.698 ± 0.587	0.914 ± 0.006	$\mathbf{0.861 \pm 0.025}$	47.8
DPS-NN	1	1.261 ± 2.340	0.735 ± 0.012	0.667 ± 0.113	109.3
	4	0.318 ± 0.770	$\mathbf{0.895 \pm 0.006}$	$\mathbf{0.873 \pm 0.023}$	277.7
SCG	8	0.214 ± 0.368	0.877 ± 0.006	0.847 ± 0.014	531.6
	16	$\mathbf{0.131 \pm 0.325}$	0.880 ± 0.003	0.842 ± 0.031	1242.6
	4	0.151 ± 0.298	$\mathbf{0.906 \pm 0.006}$	$\mathbf{0.861 \pm 0.011}$	301.9
Classifier + SCG	8	0.098 ± 0.179	0.893 ± 0.004	0.839 ± 0.024	555.6
	16	$\mathbf{0.064 \pm 0.159}$	0.899 ± 0.007	0.849 ± 0.018	1253.9

Table 5. Trade-offs between controllability, OA and computational time. n refers to number of samples at each step.

Impact of Sampling Strategy. By default, we use DDPM with 1000 steps as the base sampling algorithm, and apply SCG for rule guidance after 250 steps ($t = 750$). The reason that we do not start SCG from the beginning is that the decoded piano rolls at the beginning are quite sparse after thresholding the background. Consequently, the losses between the generated attributes and target attributes are almost the same among different realizations at this stage, making it ineffective for selecting the best directions.

To reduce the computational cost, we explore various sampling strategies, as detailed in Table 6. Firstly, we experimented with applying SCG intermittently, every k steps ($k = 2, 5$), and specifically during either the initial phase (750-400) or the latter phase (400-0) of the DDPM-1000 process. Among these variants, conducting SCG every 2 steps yielded the lowest loss. While the loss remains higher than in our default setting, this approach is about twice as fast. Additionally, we observed that applying SCG during the early phase of the process is more effective than in the later phase, likely due to greater perturbations early on, which

Method	Guided Steps	Loss ↓	OA ↑	Time (s)
DDPM-1000	750-0	**0.131 ± 0.325**	0.880 ± 0.003	1242.6
	every 2	0.365 ± 0.559	0.893 ± 0.006	635.4
	every 5	0.632 ± 0.577	0.879 ± 0.005	269.8
	750-400	0.458 ± 0.647	0.902 ± 0.009	594.7
	400-0	1.297 ± 1.772	**0.912 ± 0.007**	674.6
DDPM†-800	750-200	**0.183 ± 0.341**	**0.864 ± 0.005**	912.6
DDPM†-700	750-300	1.950 ± 1.344	0.737 ± 0.011	747.3
sDDIM-100	all	**0.303 ± 0.509**	**0.887 ± 0.005**	164.3
sDDIM-50	all	0.372 ± 0.915	0.879 ± 0.008	81.9
sDDIM-25	all	0.428 ± 0.683	0.859 ± 0.005	40.7

Table 6. Impact of sampling strategy. The numbers that follow the method names are the total sampling steps. †: early stopping.

Figure 3. Subjective evaluation scores.

enhance the likelihood of identifying optimal directions (see Appendix F for more details).

Secondly, we considered early stopping of the DDPM-1000 process after k steps ($k = 800, 700$). This is motivated by our use of post-processing techniques like thresholding and smoothing note velocity on piano rolls, which reduces the need for fine-tuning in the latter stages of the generation process. Early stopping at 200 steps resulted in only marginally inferior outcomes but cut computational time by a quarter.

Finally, we explore the compatibility of SCG with other popular sampling algorithm for diffusion models, such as DDIM (Song et al., 2021a). By default, DDIM is deterministic. However, our SCG algorithm needs stochasticity to search for the best direction. Therefore, we set stochasticity $\eta = 1$ in the DDIM algorithm and refer the modified algorithm as stochastic DDIM (sDDIM). We tested sDDIM with 100, 50 and 25 steps. More steps offers lower loss and better music quality at a cost of longer sampling time.

6.6. Subjective Evaluation

To compare performance of our SCG algorithm and baselines (classifier guidance and DPS), we carried out a listening test. We crafted four sets of rules (each set comprised of PH, ND, and CP), and use each method to generate samples that follow the rules, yielding a total of 12 samples, each 10.24 seconds long. Experienced listeners assess the quality of samples in 4 dimensions: rule alignment, musical creativity, musical coherence, and overall rating. In Figure 3, SCG consistently outperforms the baselines in all dimensions. For details of our survey, please see Appendix H.

6.7. Examples of Our System as a Compositional Tool

To demonstrate how our system can be used effectively as a compositional tool, we provide links to three example videos, available through this website. For each video, a musician first indicated desired musical characteristics in terms of the rules (e.g. fairly sparse excerpt, following a simple I-V chord progression in C major, etc). The musician's plan was to then loop this and use that as an accompaniment over
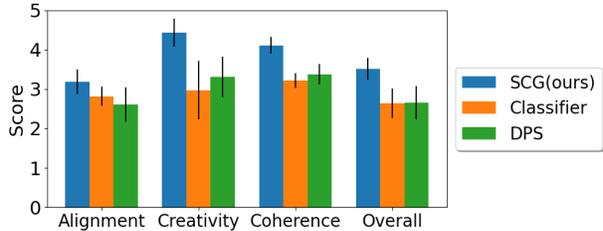
which they would then improvise. The system generated 3-5 options (i.e. samples) for each such request, and the musician chose their preferred sample with which to work, from each set. They sent the generated MIDI output to a Disklavier piano and then recorded a second track over top of that. In the accompanying videos, the musician is playing along with the generated music by our approach.

In **Video 1**, the model generated material that suggested a melody. Since the musician wanted to play the second track in the upper register, they first allowed the excerpt to play in full, as generated, and then removed the upper notes from the accompaniment to give room for themselves to play overtop. They chose to use the model's generated melody as a motif, and further improvise based on it.

In **Video 2**, the model generates an excerpt with a steady accompanying triplet ostinato behind a slower-moving descending melody in C minor (that suggests a progression that moves between the I and the V).

In **Video 3**, the model generates a sample with a changing note density and texture, and a slightly ambiguous harmonic quality that allowed flexibility in the improvising over it.

7. Conclusion and Future directions

We introduced a symbolic music generator with non-differentiable rule guided diffusion models, drawing inspiration from stochastic control. Comprehensive evaluations show our model's superiority over previous works, highlighting the potential of rule-guided approaches in enhancing creativity and control in computational music composition.

In principle, the SCG algorithm introduced in this paper extends beyond the realm of symbolic music generation. Its capability to enforce diffusion models to follow non-differentiable constraints makes it suitable for diverse fields, as long as the constraints can be programmed and one can define a loss on how well the constraints are satisfied. For instances, in protein design, one can write a function to return how specific topological constraints are satisfied. In astronomy imaging, one can use black-box physics simulators as the rule function. We believe it is an exciting future direction to extend this algorithm to a broader scope.

Acknowledgements

Impact Statement

This paper presents work whose goal is to advance the field of generative modeling for symbolic music. While there are numerous potential societal implications associated with our work, we believe none require specific emphasis in this context.

References

Atassi, L. Generating symbolic music using diffusion models. *arXiv preprint arXiv:2303.08385*, 2023.

Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Brunner, G., Konrad, A., Wang, Y., and Wattenhofer, R. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. *19th International Society for Music Information Retrieval Conference*, 2018.

Choi, J., Kim, S., Jeong, Y., Gwon, Y., and Yoon, S. Ilvr: Conditioning method for denoising diffusion probabilistic models. *ICCV*, 2021.

Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., and Engel, J. Encoding musical style with transformer autoencoders. In *International Conference on Machine Learning*, pp. 1899–1908. PMLR, 2020.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. *ICLR*, 2023.

Cuthbert, M. S. and Ariza, C. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

Dai Pra, P. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23:313–329, 1991.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., and Yang, Y.-H. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Efron, B. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Fleming, W. H. and Mitter, S. K. Optimal control and nonlinear filtering for nondegenerate diffusion processes. *Stochastics: An International Journal of Probability and Stochastic Processes*, 8(1):63–77, 1982.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., and Yang, Y.-H. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 178–186, 2021.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Huang, Q., Park, D. S., Wang, T., Denk, T. I., Ly, A., Chen, N., Zhang, Z., Zhang, Z., Yu, J., Frank, C., et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.

Huang, Y.-S. and Yang, Y.-H. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 1180–1188, 2020.

Kappen, H. J. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011, 2005.

Kawar, B., Elad, M., Ermon, S., and Song, J. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

Li, S. and Sung, Y. Melodydiffusion: Chord-conditioned melody generation using a transformer-based diffusion model. *Mathematics*, 11(8):1915, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *ICLR*, 2019.

Lu, P., Xu, X., Kang, C., Yu, B., Xing, C., Tan, X., and Bian, J. Musecoco: Generating symbolic music from text. *arXiv preprint arXiv:2306.00110*, 2023.

Meade, N., Barreyre, N., Lowe, S. C., and Oore, S. Exploring conditioning for generative music systems with human-interpretable controls. *arXiv preprint arXiv:1907.04352*, 2019.

Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. Sdedit: Image synthesis and editing with stochastic differential equations. *ICLR*, 2022.

Min, L., Jiang, J., Xia, G., and Zhao, J. Polyffusion: A diffusion model for polyphonic score generation with internal and external controls. *Proc. of the 24th Int. Society for Music Information Retrieval Conf*, 2023.

Øksendal, B. *Stochastic differential equations*. Springer, 2003.

Pavon, M. Stochastic control and nonequilibrium thermodynamical systems. *Applied Mathematics and Optimization*, 19:187–202, 1989.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Ren, Y., He, J., Tan, X., Qin, T., Zhao, Z., and Liu, T.-Y. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 1198–1206, 2020.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning*, pp. 4364–4373. PMLR, 2018.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *ICLR*, 2021a.

Song, J., Zhang, Q., Yin, H., Mardani, M., Liu, M.-Y., Kautz, J., Chen, Y., and Vahdat, A. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, 2023.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021b.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, pp. 127063, 2023.

Theodorou, E., Buchli, J., and Schaal, S. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

Theodorou, E. A. Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations. *Entropy*, 17(5):3352–3375, 2015.

Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

von Rütte, D., Biggio, L., Kilcher, Y., and Hofmann, T. Figaro: Controllable music generation using learned and expert features. In *The Eleventh International Conference on Learning Representations*, 2022.

Wang, Z., Chen, K., Jiang, J., Zhang, Y., Xu, M., Dai, S., Gu, X., and Xia, G. Pop909: A pop-song dataset for music arrangement generation. *arXiv preprint arXiv:2008.07142*, 2020.

Wu, S.-L. and Yang, Y.-H. Musemorphose: Full-song and fine-grained piano music style transfer with one transformer vae. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1953–1967, 2023.

Yang, L.-C. and Lerch, A. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *International Society for Music Information Retrieval*, 2017.

Yin, Z., Reuben, F., Stepney, S., and Collins, T. Deep learning's shallow gains: a comparative evaluation of algorithms for automatic music generation. *Machine Learning*, 112(5):1785–1822, 2023.

Zhang, C., Ren, Y., Zhang, K., and Yan, S. Sdmuse: Stochastic differential music editing and generation via hybrid representation. *IEEE Transactions on Multimedia*, 2023a.

Zhang, Q. and Chen, Y. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Zhang, Q., Song, J., Huang, X., Chen, Y., and Liu, M.-Y. Diffcollage: Parallel generation of large content with diffusion models. *CVPR*, 2023b.

A. Proofs.

A.1. Proof of Theorem 4.1.

Lemma A.1 (Dai Pra, 1991; Pavon, 1989). *The transition probability for the stochastic dynamical system Eq 6 with cost Eq 7 and optimal control $\mathbf{u}^*$ is:*

$$\mathcal{Q}_{s,t}^*(\boldsymbol{\eta}_s, \boldsymbol{\eta}_t) = \mathcal{Q}_{s,t}^0(\boldsymbol{\eta}_s, \boldsymbol{\eta}_t) \frac{\Psi(\boldsymbol{\eta}_t, t)}{\Psi(\boldsymbol{\eta}_s, s)} \tag{18}$$

where $\mathcal{Q}_{s,t}^*(\boldsymbol{\eta}_s, \boldsymbol{\eta}_t)$ denotes the transition probability from state $\boldsymbol{\eta}_s$ at time s to state $\boldsymbol{\eta}_t$ at time t, and $\mathcal{Q}_{s,t}^0(\boldsymbol{\eta}_s, \boldsymbol{\eta}_t)$ denotes the transition probability of the uncontrolled system Eq 5.

Now we prove Theorem 4.1.

Proof. Consider the SDE in Eq 6 with initial condition $\boldsymbol{\eta}_0 \sim \delta_{\bar{\boldsymbol{\eta}}_0}$, where $\delta_{\bar{\boldsymbol{\eta}}_0}$ is a Dirac distribution centered at $\bar{\boldsymbol{\eta}}_0$. Define the terminal cost to be $\phi(\boldsymbol{\eta}_T) = \log \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})}$, where $p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)$ denotes the terminal distribution of the uncontrolled SDE in Eq 5 with initial condition $\boldsymbol{\eta}_0 \sim \delta_{\bar{\boldsymbol{\eta}}_0}$, and the target terminal distribution $p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y}) := p(\mathbf{y}|\boldsymbol{\eta}_T)p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)/p(\mathbf{y})$. Then we have

$$\Psi(\bar{\boldsymbol{\eta}}_0, 0) = \mathbb{E}_{\mathcal{Q}^0}\left[e^{-\phi(\boldsymbol{\eta}_T)}|\boldsymbol{\eta}_t = \bar{\boldsymbol{\eta}}_0\right] = \int_{\boldsymbol{\eta}_T} \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)} p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T) \mathrm{d}\boldsymbol{\eta}_T = 1 \tag{19}$$

$$\Psi(\boldsymbol{\eta}_T, T) = e^{-\phi(\boldsymbol{\eta}_T)} = \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)} \tag{20}$$

Then from Lemma A.1, we have

$$\begin{aligned}
\mathcal{Q}_{0,T}^*(\bar{\boldsymbol{\eta}}_0, \boldsymbol{\eta}_T) &= \mathcal{Q}_{0,T}^0(\bar{\boldsymbol{\eta}}_0, \boldsymbol{\eta}_T) \frac{\Psi(\boldsymbol{\eta}_T, T)}{\Psi(\bar{\boldsymbol{\eta}}_0, 0)} \\
&= p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T) \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)} \\
&= p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})
\end{aligned} \tag{21}$$

From the properties of reverse-time SDE, we know that if $p_0(\boldsymbol{\eta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, then $p_T(\boldsymbol{\eta}) = p(\boldsymbol{\eta})$, i.e. $\int p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)p_0(\bar{\boldsymbol{\eta}}_0)\mathrm{d}\bar{\boldsymbol{\eta}}_0 = p(\boldsymbol{\eta}_T)$. It follows that

$$\begin{aligned}
\mathcal{Q}^*(\boldsymbol{\eta}_T) &= \int \mathcal{Q}_{0,T}^*(\bar{\boldsymbol{\eta}}_0, \boldsymbol{\eta}_T)p_0(\bar{\boldsymbol{\eta}}_0)\mathrm{d}\bar{\boldsymbol{\eta}}_0 \\
&= \int p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})p_0(\bar{\boldsymbol{\eta}}_0)\mathrm{d}\bar{\boldsymbol{\eta}}_0 \\
&= \int \frac{p(\mathbf{y}|\boldsymbol{\eta}_T)p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)}{p(\mathbf{y})}p_0(\bar{\boldsymbol{\eta}}_0)\mathrm{d}\bar{\boldsymbol{\eta}}_0 \\
&= \frac{p(\mathbf{y}|\boldsymbol{\eta}_T)}{p(\mathbf{y})}\int p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)p_0(\bar{\boldsymbol{\eta}}_0)\mathrm{d}\bar{\boldsymbol{\eta}}_0 \\
&= \frac{p(\mathbf{y}|\boldsymbol{\eta}_T)}{p(\mathbf{y})}p(\boldsymbol{\eta}_T) \\
&= p(\boldsymbol{\eta}_T|\mathbf{y})
\end{aligned} \tag{22}$$

Finally, we show that the optimal control for $\tilde{\phi} = \ell_y(\boldsymbol{\eta}_T)$ is the same as that for $\phi(\boldsymbol{\eta}_T) = \log \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})}$, because

$$\phi(\boldsymbol{\eta}_T) = \log \frac{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T)}{p_{\bar{\boldsymbol{\eta}}_0}(\boldsymbol{\eta}_T|\mathbf{y})} = \log \frac{p(\mathbf{y})}{p(\mathbf{y}|\boldsymbol{\eta}_T)} = -\log p(\mathbf{y}|\boldsymbol{\eta}_T) + const = \ell_y(\boldsymbol{\eta}_T) + const \tag{23}$$

The last equality follows from our assumption that $p_0(\mathbf{y}|\boldsymbol{\eta}) \propto e^{-\ell_y(\boldsymbol{\eta})}$. Plugging $\tilde{\phi}(\boldsymbol{\eta}_T)$ and $\phi(\boldsymbol{\eta}_T)$ into Eq 13 leads to the same optimal control. $\square$

A.2. Proof of Proposition 4.2

Proof.

$$\Psi(\boldsymbol{\eta}, t) = \mathbb{E}_{\mathcal{Q}^0}\left[e^{-\phi(\boldsymbol{\eta}_T)}|\boldsymbol{\eta}_t = \boldsymbol{\eta}\right]$$
$$= \mathbb{E}_{\mathcal{Q}^0}\left[p(\mathbf{y}|\boldsymbol{\eta}_T) \cdot c|\boldsymbol{\eta}_t = \boldsymbol{\eta}\right]$$
$$= c \cdot \int p(\mathbf{y}|\boldsymbol{\eta}_T)p(\boldsymbol{\eta}_T|\boldsymbol{\eta}_t = \boldsymbol{\eta})\mathrm{d}\boldsymbol{\eta}_T \tag{24}$$

where $p(\boldsymbol{\eta}_T|\boldsymbol{\eta}_t = \boldsymbol{\eta}) := \mathcal{Q}_{t,T}^0(\boldsymbol{\eta}, \boldsymbol{\eta}_T)$ is the transition probability from state $\boldsymbol{\eta}$ at time t to state $\boldsymbol{\eta}_T$ at time T for the uncontrolled SDE, and it is differentiable with respect to $\boldsymbol{\eta}$ for all $0 \le t < T$. In addition, notice that in diffusion models,

$$p(\mathbf{y}|\boldsymbol{\eta}_t) = \int p(\mathbf{y}|\boldsymbol{\eta}_T, \boldsymbol{\eta}_t)p(\boldsymbol{\eta}_T|\boldsymbol{\eta}_t)\mathrm{d}\boldsymbol{\eta}_T$$
$$= \int p(\mathbf{y}|\boldsymbol{\eta}_T)p(\boldsymbol{\eta}_T|\boldsymbol{\eta}_t)\mathrm{d}\boldsymbol{\eta}_T \tag{25}$$

where the second equality comes from that $\boldsymbol{\eta}_t$ and $\mathbf{y}$ are conditionally independent given $\boldsymbol{\eta}_T$. Then from Eq 24, we have $\Psi(\boldsymbol{\eta}_t, t) = c \cdot p(\mathbf{y}|\boldsymbol{\eta}_t)$. $\qquad\square$

Remark A.2. Although $\phi(\boldsymbol{\eta}_T)$ could be non-differentiable when we choose non-differentiable loss functions, the desirability function $\Psi(\boldsymbol{\eta}, t)$ is differentiable with respect to $\boldsymbol{\eta}$ for all $0 \le t < T$.

B. Compatibility of SCG with Various Sampling Procedures

SCG is compatible with many stochastic sampling procedures in diffusion models. The key of SCG is to sample multiple $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$, and select the one that leads to the sample that follows the rule best. One can choose different sampling procedure to obtain $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$. Algorithm 2 shows how to use SCG with replacement-based editing. Algorithm 3 shows how to use SCG with stochastic DDIM (Song et al., 2021a).

Algorithm 2 Editing with SCG.

Require: Encoding of the source music $\tilde{\mathbf{x}}_0$, mask $\mathbf{M}$ (1 for unaltered part and 0 for editing region), noise level K, sampling algorithm (e.g. SCG), desired label $\mathbf{y}$ (optional, do not need if want to create a variant).
$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\mathbf{x}_K = \sqrt{\bar{\alpha}_K}\tilde{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_K}\mathbf{z}$
for $t = K$ **to** 1 **do**
$\quad \triangleright$ Estimate the clean sample from noisy sample.
$\quad \hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta\left(\mathbf{x}_t, t\right)\right)$
$\quad \triangleright$ Replacement projection based on the mask.
$\quad \tilde{\mathbf{x}}_0 = \mathbf{M} \odot \mathbf{x}_0 + (\mathbf{I} - \mathbf{M}) \odot \hat{\mathbf{x}}_0$
$\quad \triangleright$ Predict ϵ from $\tilde{\mathbf{x}}_0$.
$\quad \tilde{\epsilon} = \frac{1}{\sqrt{1 - \bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\tilde{\mathbf{x}}_0)$
$\quad \triangleright$ Sampling using corrected ϵ.
$\quad \mathbf{x}_{t-1} = \text{sampling\_algorithm}(\mathbf{x}_t, t, \epsilon, \mathbf{y})$
end for
return: $\mathbf{x}_0$

C. Additional Experiment Results

C.1. Unconditional Generation

In Table 7, we report the overlapping area between the intra-set and inter-set distribution for all 7 musical attributes as proposed in (Yang & Lerch, 2020). The highest and second highest value except for GT are high- lighted in bold and underline respectively. Our method achieves the highest average OA on all the datasets, and achieves the top 2 OA for most of the individual attributes.

Algorithm 3 Stochastic Control Guided stochastic DDIM sampling

---

Require: Loss function ℓ_y, rule target $\mathbf{y}$, number of samples n, stochasticity $\eta > 0$, number of steps S.

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $s = S$ **to** 1 **do**

   ▷ Compute the posterior mean of $\mathbf{x}_{\tau_{s-1}}$.

   $\sigma_{\tau_s} = \eta \sqrt{\frac{1 - \bar{\alpha}_{\tau_{s-1}}}{1 - \bar{\alpha}_{\tau_s}} \left(1 - \frac{\bar{\alpha}_{\tau_s}}{\bar{\alpha}_{\tau_{s-1}}} \right)}$

   $\hat{\mathbf{x}}_{\tau_{s-1}} = \sqrt{\bar{\alpha}_{\tau_{s-1}}} \left(\frac{\mathbf{x}_{\tau_s} - \sqrt{1 - \bar{\alpha}_{\tau_s}} \epsilon_\theta(\mathbf{x}_{\tau_s}, \tau_s)}{\sqrt{\bar{\alpha}_{\tau_s}}} \right) + \sqrt{1 - \bar{\alpha}_{\tau_{s-1}} - \sigma_{\tau_s}^2} \, \epsilon_\theta(\mathbf{x}_{\tau_s}, \tau_s)$

   if $s > 1$ **then**

      ▷ Sampling possible next steps.

      $\mathbf{x}_{\tau_{s-1}}^i = \hat{\mathbf{x}}_{\tau_{s-1}} + \sigma_{\tau_s} \mathbf{z}^i$, with $\mathbf{z}^1, ..., \mathbf{z}^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

      ▷ Estimate the clean sample from noisy sample.

      $\hat{\mathbf{x}}_0^i = \frac{1}{\sqrt{\bar{\alpha}_{\tau_{s-1}}}} \left(\mathbf{x}_{\tau_{s-1}}^i - \sqrt{1 - \bar{\alpha}_{\tau_{s-1}}} \epsilon_\theta \left(\mathbf{x}_{\tau_{s-1}}^i, \tau_{s-1} \right) \right)$

      ▷ Find the direction that minimizes the loss.

      $k = \text{argmax}_i \log p(\mathbf{y}|\hat{\mathbf{x}}_0^i) = \text{argmax}_i -\ell_y(\hat{\mathbf{x}}_0^i)$

      $\mathbf{x}_{\tau_{s-1}} = \mathbf{x}_{\tau_{s-1}}^k$

   else

      $\mathbf{x}_{\tau_{s-1}} = \hat{\mathbf{x}}_{\tau_{s-1}}$

   end if

end for

return: $\mathbf{x}_0$

In addition, we check if the models are copying from the dataset. To do so, we randomly pick 50 generated samples for each method and 2000 samples from the training dataset, and identify the closest musical piece to a generated MIDI file from the dataset. Specifically, we extract and compare key features from each file, including pitch, velocity, duration of notes, and overall note density. We report the average matching score of these four features in Table 8. As we can see, despite our method achieving the highest OA, it has the second lowest matching score, indicating that the OA improvement is not from copying from the dataset.

C.2. Editing

Our framework also supports editing. Given an existing music piece, we can modify it within any given time window: either create a new variant or guide it to satisfy new rules. To achieve this, we mainly follow the SDEdit framework (Meng et al., 2022): first we add Gaussian noise of a chosen noise level to the latent music representation and then progressively remove the noise by reversing the SDE. During the reverse process, we use a mask to distinguish the parts that we want to preserve unaltered and the portion we want to modify, and we condition on the unaltered parts via replacement-based conditioning methods as in (Choi et al., 2021; Kawar et al., 2022). Please refer to Appendix B for the detailed guided editing algorithm.

We benchmark our music editing performance against two establibled methods: MuseMorphose (Wu & Yang, 2023) and PolyDiffusion (Min et al., 2023). Since these baselines are trained on Pop piano music, we condition on the Pop dataset when evaluating our method. Unlike these baselines, which restrict editing to one specific attribute, our method offers the flexibility to edit any attribute. The editing task involves creating a new music piece that adheres to predefined rules based on an original source music piece (for detailed settings, see Appendix D.3). To assess controllability, we evaluate the error rate between the target and generated attributes. Additionally, we measure the similarity in chroma and groove between the generated piece and the source to gauge their resemblance. The goal is to generate music that not only complies with the desired rules but also closely resembles the original source music.

Table 9 shows the results. For note density, we experiment with two noise levels: 400 and 500. For chord progression, we use a noise level of 500. We can see that there is a trade-off between controllability and resemblance: higher noise level results in better controllability (lower error) but reduced resemblance (lower similarity metrics).

Dataset	Method	Used Pitch	IOI	Pitch Hist	Pitch Range	Velocity	Note Duration	Note Density	Avg
Maestro	GT	0.960 ± 0.007	0.901 ± 0.007	0.980 ± 0.003	0.962 ± 0.004	0.971 ± 0.004	0.884 ± 0.011	0.953 ± 0.005	0.944 ± 0.002
	MusicTr	0.954 ± 0.004	0.896 ± 0.018	0.948 ± 0.011	0.961 ± 0.005	0.967 ± 0.005	0.647 ± 0.022	0.948 ± 0.007	0.903 ± 0.005
	Remi	0.934 ± 0.018	0.794 ± 0.017	0.897 ± 0.010	0.903 ± 0.015	0.906 ± 0.007	0.542 ± 0.021	0.952 ± 0.004	0.847 ± 0.005
	CPW	0.941 ± 0.012	0.641 ± 0.025	0.866 ± 0.010	0.962 ± 0.005	0.830 ± 0.014	0.436 ± 0.030	0.933 ± 0.007	0.801 ± 0.006
	PolyDiff	0.852 ± 0.006	0.843 ± 0.026	0.888 ± 0.016	0.871 ± 0.006	0.805 ± 0.031	0.777 ± 0.016	0.856 ± 0.005	0.842 ± 0.007
	Ours	**0.961 ± 0.006**	**0.901 ± 0.009**	**0.960 ± 0.010**	**0.963 ± 0.006**	**0.971 ± 0.004**	**0.910 ± 0.012**	0.934 ± 0.005	**0.943 ± 0.003**
Muscore	GT	0.959 ± 0.009	0.928 ± 0.019	0.980 ± 0.005	0.965 ± 0.006	0.896 ± 0.008	0.925 ± 0.008	0.963 ± 0.004	0.945 ± 0.004
	MusicTr	0.952 ± 0.012	0.916 ± 0.008	0.891 ± 0.009	0.958 ± 0.005	0.790 ± 0.008	0.851 ± 0.020	**0.949 ± 0.005**	0.901 ± 0.004
	Remi	0.924 ± 0.008	**0.917 ± 0.016**	0.955 ± 0.010	0.944 ± 0.007	0.731 ± 0.024	0.748 ± 0.028	0.934 ± 0.003	0.879 ± 0.006
	CPW	0.879 ± 0.012	0.839 ± 0.020	0.941 ± 0.008	0.900 ± 0.006	0.750 ± 0.025	0.715 ± 0.030	0.879 ± 0.010	0.843 ± 0.007
	PolyDiff	0.888 ± 0.009	0.831 ± 0.006	0.927 ± 0.012	0.864 ± 0.007	0.693 ± 0.014	0.822 ± 0.015	0.891 ± 0.008	0.845 ± 0.004
	Ours	**0.963 ± 0.004**	0.915 ± 0.009	**0.962 ± 0.009**	**0.964 ± 0.004**	**0.890 ± 0.006**	**0.900 ± 0.012**	0.946 ± 0.005	**0.934 ± 0.003**
Pop	GT	0.956 ± 0.007	0.949 ± 0.006	0.983 ± 0.002	0.955 ± 0.003	0.954 ± 0.004	0.940 ± 0.009	0.963 ± 0.002	0.957 ± 0.002
	MusicTr	0.807 ± 0.016	0.880 ± 0.010	0.852 ± 0.005	0.833 ± 0.011	0.865 ± 0.014	0.871 ± 0.008	0.809 ± 0.011	0.845 ± 0.004
	Remi	0.870 ± 0.014	0.839 ± 0.007	**0.979 ± 0.002**	0.827 ± 0.008	0.853 ± 0.013	0.826 ± 0.012	0.867 ± 0.005	0.866 ± 0.004
	CPW	0.921 ± 0.011	0.803 ± 0.022	0.942 ± 0.010	0.927 ± 0.008	0.853 ± 0.006	0.891 ± 0.011	**0.953 ± 0.008**	0.899 ± 0.005
	PolyDiff	**0.941 ± 0.003**	0.924 ± 0.012	0.964 ± 0.005	**0.937 ± 0.006**	0.648 ± 0.007	**0.912 ± 0.020**	0.855 ± 0.012	0.883 ± 0.004
	Ours	0.927 ± 0.009	**0.952 ± 0.004**	0.969 ± 0.002	0.928 ± 0.013	**0.948 ± 0.003**	0.911 ± 0.019	0.941 ± 0.009	**0.939 ± 0.004**

Table 7. Objecetive evaluation of unconditional generation. The overlapping area (OA) for 7 music attributes and the average OA are reported. The highest and second highest OA excluding GT are bolded and underlined respectively.

Method	Matching Score	OA
MusicTr	0.6273 ± 0.0336	0.903 ± 0.005
Remi	0.6386 ± 0.0210	0.866 ± 0.004
CPW	0.6439 ± 0.0134	0.899 ± 0.005
PolyDiff	0.6372 ± 0.0097	0.883 ± 0.004
Ours	0.6337 ± 0.0185	0.943 ± 0.003

Table 8. Test if the model is copying samples from the dataset by evaluating Matching Score and OA.

D. Detailed Experiment Setup

D.1. Music Rules

We consider three music rules and give their definitions below.

Pitch Histogram: We compute the histogram of 12 pitch classes over the whole piano roll. Pitch velocity is considered when computing the histogram. The target $\mathbf{y}$ is a 12-dimensional vector specifying the desired pitch histogram.

Note Density: We control both vertical and horizontal note density. We compute note density within 128×128 windows. For a piano roll of shape 128×1024, the target $\mathbf{y}$ is a 16-dimensional vector, the first 8 dimension are for vertical note density and the last 8 dimension are for horizontal note density.

Vertical note density $\mathrm{ND}_{\mathrm{vertical}}$ is computed by

$$\mathrm{ND}_{\mathrm{vertical}} = \frac{1}{T}\sum_{t=1}^{T} n_{\mathrm{on}}(t) \tag{26}$$

where $n_{\mathrm{on}}(t)$ stands for the number of on-notes at time t, and T is the window size, we set $T = 128$.

Horizontal note density $\mathrm{ND}_{\mathrm{horizontal}}$ is computed by

$$\mathrm{ND}_{\mathrm{horizontal}} = \sum_{t=1}^{T} \mathbb{1}(n_{\mathrm{start}}(t) \geq 1) \tag{27}$$

where $n_{\mathrm{start}}(t)$ stands for the number of notes that start at time t.

Chord Progression: We extract chords using chord analysis tool from the music21 (Cuthbert & Ariza, 2010) package, and group them into 7 classes. We extract 8 chords in total for the 128×1024 piano roll, each chord is the longest chord within a 128×128 window. The target $\mathbf{y}$ is an 8-dimensional vector specifying the desired chord for each 128×128 window.

Rule	Method	Error (%) $\downarrow$	$Sim_{chr} \uparrow$	$Sim_{grv} \uparrow$
Note Density	MuseMorphose	29.34	**0.9130**	**0.9184**
	Ours-400	35.62	0.9119	0.8511
	Ours-500	**27.87**	0.8173	0.7153
Chord Progression	PolyDiffusion	70.48	0.5902	**0.7515**
	Ours	**12.62**	**0.8236**	0.6974

Table 9. Editing performance. For note density, we experimented with noise level of 400 and 500. For chord progression, we used noise level of 500.

D.2. Training Setup

Data Augmentation. We use the same data augmentation for both VAE and diffusion model training.

- **Key shift**: Entire piano rolls were shifted by up to 6 pitches, effectively functioning as a key switch.

- **Time shift**: We load in a piano roll of 2 times the desired length, and randomly select a starting time to obtain the actual piano roll for training.

- **Tempo shift**: Tempo of the piece was shifted by a factor of [0.95, 1.05].

VAE. Utilizing the standard autoencoder architecture from (Rombach et al., 2022), we compressed piano roll segments (dimension $3 \times 128 \times 128$) into a latent space of $4 \times 16 \times 16$. The three dimensions of the piano roll include onset and pedal information, in addition to the standard piano roll data.

Let x represent the piano roll in pixel space and z the latent code. We denote the encoder and decoder as $\mathcal{E}$ and $\mathcal{D}$, respectively. The training objective for our VAE model is formulated as follows:

$$\mathcal{L}_{VAE} = \|\mathcal{D}(\mathcal{E}(x)) - x\|_1 + \lambda_{\text{KL}}(t)D_{KL}(\mathcal{N}(z;\mathcal{E}_\mu,\mathcal{E}_{\sigma^2})\|\mathcal{N}(z;0,I)) + \lambda_{\text{denoise}}(t)\|\mathcal{D}(\mathcal{E}(\text{Noisy}(x))) - x\|_1 \quad (28)$$

The first term is the standard reconstruction loss, where we used L1 loss to encourage sparsity. The second term is the standard KL regularization term weighted by a scheduler $\lambda_{\text{KL}}(t)$. The third term is a denoising reconstruction loss, influenced by the scheduler $\lambda_{\text{denoise}}(t)$. Here, Noisy refers to a perturbation operator applied to the piano roll, encompassing:

- **Note shift**: Some fraction of notes were randomly selected by uniform distribution to be perturbed. Perturbed notes were shifted by up to 2 pitches higher or lower.

- **Adjacent note addition**: Some fraction of notes were randomly selected by uniform distribution. A second identical note was added just one pitch higher or lower to the original note. These adjacent notes are quite discordant to the ear.

- **Rhythm shift**: Some fraction of notes were randomly selected by uniform distribution to be perturbed. Perturbed notes were shifted by up to 100 ms earlier or later.

- **Note deletion**: Some fraction of notes were randomly selected by uniform distribution to be deleted.

We capped the maximum fraction of perturbed notes at 25% for all perturbations. The model was trained over 800k steps. The KL scheduler, $\lambda_{\text{KL}}(t)$, was a linear scheduler increasing from 0 to $1e-2$ across 400k steps. The denoising scheduler, λ_{denoise}, linearly increased the perturbation fraction from 0 to 25% over 400k steps. We employed a cosine learning rate scheduler with a 10k-step warmup, peaking at a learning rate of $5e-4$. The optimizer used was AdamW (Loshchilov & Hutter, 2019), with weight decay of 0.01 and a batch size of 80.

Diffusion Model. We train our diffusion model with a transformer backbone on the latent space of a pretrained VAE. First we rescale the latent representation $\mathcal{E}(x)$ by its standard deviation, computed using a batch of 256 training samples as per the methodology described in (Rombach et al., 2022). Then we reshaped the latent representation from $4 \times 16 \times 128$ to 32×256, followed by a transformation of the 32-dimensional vector to match the hidden dimension of the transformer backbone. We employed the DiT-XL architecture from (Peebles & Xie, 2023), which has a hidden dimension of 1152.

In addition, we use rotary positional embedding (Su et al., 2023) to accommodate for various length of input (e.g. when generating longer sequence of music).

Given our use of data augmentation during diffusion model training, it was necessary to compute $\mathcal{E}(x)$ dynamically, a process that is typically time-consuming. To optimize this, we employed a strategy to avoid encoding each sample from scratch. During data loading, we initially loaded a piano roll of length 2560 and then encoded each 128-length segment using the pretrained encoder, resulting in 20 latent codes. By concatenating subsets of these latent codes, we generated 4 training samples (segments 1-8, 5-12, 9-16, and 13-20), each measuring $8 \times 128 = 1024$ in length.

We train our model on three datasets using the training procedure of classifier-free guidance (Ho & Salimans, 2022). Specifically, we set $y = 0$ for Maestro, $y = 1$ for Muscore and $y = 2$ for Pop. We jointly train a conditional model $\epsilon_\theta(\mathbf{x}_t, t, y)$ and an unconditional model $\epsilon_\theta(\mathbf{x}_t, t, \text{null})$ with a dropout rate of 0.1.

We adhered to the training hyper-parameters outlined in (Peebles & Xie, 2023). Specifically, we used a constant learning rate of $1e - 4$, no weight-decay and a batch size of 256 with the AdamW optimizer (Loshchilov & Hutter, 2019). We use linear noise scheduling and trained the model for 1.2M steps.

D.3. Objective Evaluation Setup

Unconditional Generation. In our study, we generated 400 music segments, each lasting 10.24 seconds, for all the methods under consideration. For baseline methods that utilize bars as the time unit, we produced 8-bar sequences from which we randomly extracted segments of 10.24 seconds in duration. We used the official released models for Remi (Huang & Yang, 2020), CPW (Hsiao et al., 2021) and PolyDiff (Min et al., 2023). Unfortunately, an official implementation for the music transformer (Huang et al., 2018) was not available. Consequently, we resorted to an unofficial implementation[1] and trained a music transformer by ourselves.

The overlapping area (OA) for seven music attributes was computed following the methodology described in (Yang & Lerch, 2020). To accurately evaluate OA, it is typically required that the two datasets being compared contain an equal amount of data. Therefore, we randomly selected 400 samples from the test dataset to align with the number of generated samples. This evaluation process was repeated five times to calculate the mean and standard deviation of the results in Table 2.

Individual Rule Guidance. For each rule, we randomly selected 200 samples from the Muscore test dataset and computed their corresponding rule labels to serve as targets. Subsequently, we generated 200 samples conditioned on each rule label. The rule labels for these generated samples were computed, and the loss between the generated rule label and the target was calculated. For pitch histogram and note density, we use MSE loss. For chord progression, we use 0-1 loss. The mean and standard deviation of this loss across the 200 samples are presented in Table 3.

We also compute OA between the generated samples and the data that comply with the desired rule. Specifically, we cluster 200 generated samples into 4 groups based on note density and compute OA within each group. The mean and standard deviations of OA across the 4 groups are also presented in Table 3.

Composite Rule Guidance. We randomly selected 200 samples from the Muscore test dataset and compute their rule labels for each of the three rules under consideration. Then we generated 200 samples conditioned on all three rule labels simultaneously, with the intention that the generated samples adhere to all three rules concurrently.

Ablation Studies. For all the ablation studies, we follow the individual rule guidance set up and guide the diffusion model to generate music following given note density. The computational time is for generating 4 samples in a batch. Regarding the quality metric, we randomly chose 200 samples from the test dataset and calculated the average OA similar to the process used for unconditional generation. This procedure was repeated five times to calculate the mean and standard deviation.

Editing. To facilitate a comparison with MuseMorphase (Wu & Yang, 2023), we adopted their approach for computing the note density label. Initially, we randomly selected 200 samples from the Pop test dataset and calculated both vertical and horizontal note density vectors for each sample, using a window size of 1.28 seconds. Consequently, for each 10.24-second sample, we obtained 8 vertical and 8 horizontal note density values. We then flattened these note density vectors and categorized them into 8 bins, ensuring approximately an equal number of samples in each bin for both vertical and horizontal densities. During generation, we randomly chose a shift value of -1, 0, or 1 to adjust the note density classes of a sample, using the center value of the resultant bin as the target note density.

---

[1] Available at https://github.com/gwinndr/MusicTransformer-Pytorch

We also considered PolyDiff (Min et al., 2023) as another baseline. In their approach, a new musical piece is generated based on the piano roll with basic chords extracted from the current piece, which is seen as an editing task since it creates a variation of the existing music with the same chord progression. In our framework, we extracted chords from the source music, added noise to the source, and then generated new music guided by the extracted chords.

For both baseline methods, we generated 8-bar music segments and extracted rule labels for each bar. In contrast, our method involved generating 10.24-second music segments and using a 1.28-second time window to extract rule labels, thereby aligning the number of rule labels across all methods.

In terms of similarity metrics, we calculated chroma and grooving similarity between the generated samples and their respective source samples, following the methodology outlined in (Wu & Yang, 2023).

E. Training Surrogate Models for Music Rules

For classifier guidance (Dhariwal & Nichol, 2021) and DPS-NN (Chung et al., 2023), we need to train surrogate models to approximate various music rules. We used the DiT-S architecture in (Peebles & Xie, 2023) as the backbone for classifiers [2]. Following the ViT approach (Dosovitskiy et al., 2021), we appended a class token to the latent codes and utilized a multi-layer perceptron (MLP) for the classification head. For rules like pitch histogram and note density, our classifier produces a vector of corresponding rules, and we train it using L2 loss. For chord progression, we incorporated two classifier heads: one to predict the key logits and the other for chord logits for each 128-length segment. We treated key and chord as categorical variables and trained the model using cross-entropy loss.

Figure 4 illustrates the training and validation loss/accuracy for the three rules under study. Notably, training a surrogate model for chord progression proved to be particularly challenging, with the final accuracy hovering around only 33%. This lower accuracy partly accounts for the diminished performance of rule-guided methods that depend on surrogate models.
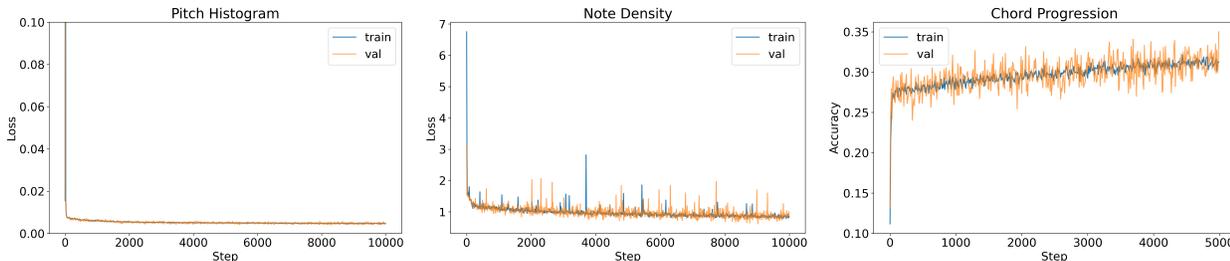

Figure 4. Training and validation curves of the classifiers trained on various rules.

F. Losses over Stochastic Control Guided Sampling Process

We recorded the lowest loss and the variation in losses at each step throughout the sampling process of a representative sample using SCG, with note density as the conditioning rule. As depicted in Figure 5 (a), we observed that the loss remains consistent until approximately $t = 750$. This early-stage constancy is attributed to the fact that, initially, the decoded piano rolls are essentially empty following the background thresholding, leading to a zero note density and, consequently, a stable loss. However, as the decoded piano rolls begin to populate, various realizations yield different note densities, resulting in a diversity of losses. By selecting the lowest loss at every step, we achieved a decrease in overall loss.

Figure 5 (b) illustrates the range of the losses at each step. The largest range occurs around $t = 750$, the point where the piano roll starts to gain semantic meaning and the best loss drops drastically. This suggests that applying guidance early, soon after the piano roll acquires semantic content, is crucial for successful guidance.

---

[2]We use classifiers to refer to the surrogate models, even if they are not necessarily trained using a classification objective
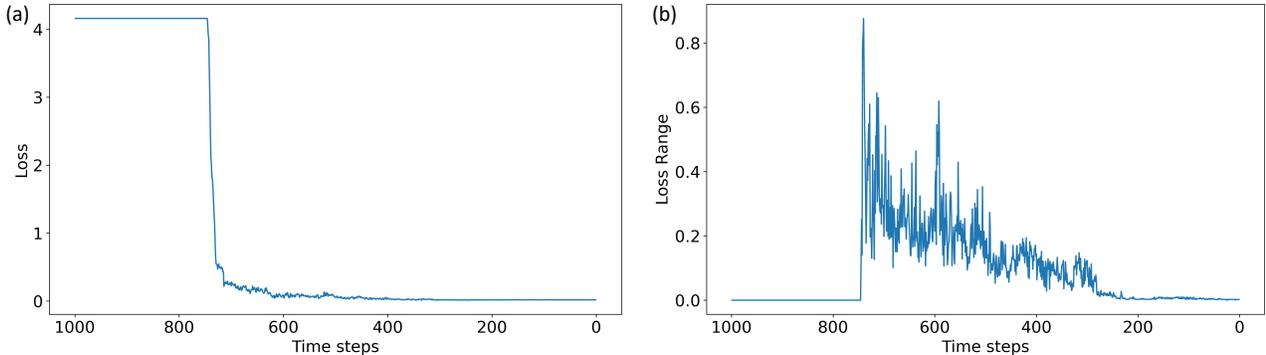
Figure 5. Best loss (a) and loss range (b) over stochastic control guided DDPM sampling on a representative sample with note density as the conditioning rule.

G. More Ablation Studies

G.1. Unconditional Generation

Our model is capable of generating samples reflective of the distributions from three distinct datasets. This is accomplished through classifier-free guidance [3] (Ho & Salimans, 2022), with conditioning based on the specific dataset. We tuned the strength of the classifier-free guidance for each dataset and discovered optimal settings for achieving the highest music quality (Table 10). Specifically, for the Maestro and Pop datasets, a guidance strength of $\omega = 0$ yielded the best results. In contrast, for the Muscore dataset, setting $\omega = 4$ proved to be most effective in enhancing musical quality.

Dataset	w	Used Pitch	IOI	Pitch Hist	Pitch Range	Velocity	Note Duration	Note Density	Avg
Maestro	0	**0.961 ± 0.006**	0.901 ± 0.009	0.960 ± 0.010	**0.963 ± 0.006**	**0.971 ± 0.004**	**0.910 ± 0.012**	0.934 ± 0.005	**0.943 ± 0.003**
	1	0.948 ± 0.002	**0.915 ± 0.011**	0.946 ± 0.004	0.931 ± 0.009	0.956 ± 0.008	**0.910 ± 0.007**	0.937 ± 0.008	0.935 ± 0.003
	2	0.953 ± 0.004	0.878 ± 0.006	**0.964 ± 0.003**	0.946 ± 0.006	0.963 ± 0.008	0.892 ± 0.009	**0.953 ± 0.004**	0.935 ± 0.001
	4	0.925 ± 0.005	0.891 ± 0.008	0.940 ± 0.007	0.934 ± 0.011	0.958 ± 0.010	0.890 ± 0.010	0.932 ± 0.005	0.924 ± 0.001
Muscore	0	0.941 ± 0.003	0.890 ± 0.019	0.950 ± 0.014	0.946 ± 0.006	0.886 ± 0.010	**0.922 ± 0.011**	0.925 ± 0.006	0.923 ± 0.003
	1	0.956 ± 0.008	0.870 ± 0.014	0.942 ± 0.007	0.962 ± 0.009	0.885 ± 0.006	0.911 ± 0.007	0.934 ± 0.007	0.923 ± 0.003
	2	0.942 ± 0.009	0.899 ± 0.014	0.940 ± 0.014	0.954 ± 0.008	0.884 ± 0.014	0.911 ± 0.007	0.924 ± 0.008	0.922 ± 0.005
	4	**0.963 ± 0.004**	**0.915 ± 0.009**	**0.962 ± 0.009**	**0.964 ± 0.004**	**0.890 ± 0.006**	0.900 ± 0.012	**0.946 ± 0.005**	**0.934 ± 0.003**
Pop	0	0.927 ± 0.009	**0.952 ± 0.004**	0.969 ± 0.002	0.928 ± 0.013	**0.948 ± 0.003**	0.911 ± 0.019	0.941 ± 0.009	**0.939 ± 0.004**
	1	0.921 ± 0.004	0.917 ± 0.003	**0.975 ± 0.003**	0.937 ± 0.009	0.939 ± 0.007	**0.926 ± 0.015**	0.935 ± 0.005	0.936 ± 0.003
	2	0.929 ± 0.008	0.885 ± 0.010	0.970 ± 0.005	0.926 ± 0.013	0.935 ± 0.007	**0.926 ± 0.012**	0.950 ± 0.011	0.932 ± 0.003
	4	**0.933 ± 0.011**	0.897 ± 0.004	0.965 ± 0.007	0.920 ± 0.010	0.940 ± 0.007	0.914 ± 0.015	**0.962 ± 0.007**	0.933 ± 0.005

Table 10. Unconditional generation on three datasets with different classifier-free guidance strength.

G.2. Latent vs Pixel Space

Our approach employed a latent diffusion model for symbolic music generation and compared its performance with a diffusion model trained in pixel space. The pixel space model was configured with a time resolution of 0.08 seconds per column in the piano roll, as opposed to the 0.01-second resolution in latent space. This choice was primarily driven by computational constraints; a 0.01-second resolution for a 10.24-second music piece would result in a piano roll of size $3 \times 128 \times 1024$, posing significant computational demands. In contrast, a 0.08-second resolution yields a more manageable size of $3 \times 128 \times 128$. For training the pixel space diffusion model, we utilized a standard U-Net backbone.

Table 11 presents a comparison of the models in the task of unconditional generation. An intriguing trend emerged: the latent space model excelled with complex, dynamic-rich music (e.g., Maestro), whereas the pixel space model showed superior performance with simpler music (e.g., Pop). The Muscore dataset, predominantly featuring classical sheet music, sits between Maestro and Pop in terms of complexity, and here, both models performed comparably. This observation aligns

---

[3] Despite this approach, we refer to it as 'unconditional generation' because it does not involve rule-based guidance.

with the notion that time resolution has a less pronounced impact on the expressiveness of simpler music, making a lower resolution viable for training the diffusion model.

dataset	method	Used Pitch	IOI	Pitch Hist	Pitch Range	Velocity	Note Duration	Note Density	Avg
Maestro	pixel	0.919 ± 0.005	0.877 ± 0.018	$\mathbf{0.983 \pm 0.005}$	0.959 ± 0.007	0.969 ± 0.003	0.897 ± 0.013	0.896 ± 0.003	0.929 ± 0.006
	latent	$\mathbf{0.961 \pm 0.006}$	$\mathbf{0.901 \pm 0.009}$	0.960 ± 0.010	$\mathbf{0.963 \pm 0.006}$	$\mathbf{0.971 \pm 0.004}$	$\mathbf{0.910 \pm 0.012}$	$\mathbf{0.934 \pm 0.005}$	$\mathbf{0.943 \pm 0.003}$
Muscore	pixel	0.962 ± 0.005	0.903 ± 0.009	$\mathbf{0.965 \pm 0.009}$	$\mathbf{0.964 \pm 0.007}$	$\mathbf{0.893 \pm 0.007}$	$\mathbf{0.928 \pm 0.011}$	0.926 ± 0.008	$\mathbf{0.934 \pm 0.005}$
	latent	$\mathbf{0.963 \pm 0.004}$	$\mathbf{0.915 \pm 0.009}$	0.962 ± 0.009	$\mathbf{0.964 \pm 0.004}$	0.890 ± 0.006	0.900 ± 0.012	$\mathbf{0.946 \pm 0.005}$	$\mathbf{0.934 \pm 0.003}$
Pop	pixel	$\mathbf{0.935 \pm 0.011}$	$\mathbf{0.957 \pm 0.004}$	$\mathbf{0.976 \pm 0.003}$	$\mathbf{0.952 \pm 0.004}$	0.945 ± 0.006	$\mathbf{0.935 \pm 0.011}$	$\mathbf{0.946 \pm 0.013}$	$\mathbf{0.949 \pm 0.005}$
	latent	0.927 ± 0.009	0.952 ± 0.004	0.969 ± 0.002	0.928 ± 0.013	$\mathbf{0.948 \pm 0.003}$	0.911 ± 0.019	0.941 ± 0.009	0.939 ± 0.004

Table 11. Comparing pixel vs latent space for unconditional generation.

Table 12 shows the loss for individual rule guidance using the pixel space-trained diffusion model. Mirroring the findings in Table 3, SCG consistently achieved the lowest loss, underscoring its effectiveness in rule guidance. However, a noticeable decline in music quality (measured by OA) was observed for the model trained on pixel space, particularly in aspects like pitch histogram and note density (Table 13). This decline can be attributed to the nature of Gaussian noise addition in pixel space, which often results in random, musically nonsensical notes that nevertheless align with rule targets. Conversely, noise addition in latent space tends to induce more meaningful alterations, thereby preserving the higher music quality.

Method	Pitch Histogram $\downarrow$	Note Density $\downarrow$	Chord Progression $\downarrow$
No Guidance	0.019 ± 0.011	2.367 ± 2.933	0.841 ± 0.142
Classifier	0.020 ± 0.015	0.287 ± 0.330	0.783 ± 0.208
DPS - NN	0.020 ± 0.013	0.615 ± 1.188	0.788 ± 0.170
DPS - Rule	0.002 ± 0.006	2.349 ± 3.425	-
SCG	$\mathbf{0.0001 \pm 0.0008}$	$\mathbf{0.103 \pm 0.570}$	$\mathbf{0.344 \pm 0.212}$

Table 12. Loss between the target and the generated attributes for individual rule guidance using the pixel-space trained diffusion model.

Model	Pitch Histogram $\uparrow$	Note Density $\uparrow$	Chord Progression $\uparrow$
Pixel	0.848 ± 0.005	0.797 ± 0.005	$\mathbf{0.892 \pm 0.009}$
Latent	$\mathbf{0.897 \pm 0.006}$	$\mathbf{0.880 \pm 0.003}$	0.883 ± 0.002

Table 13. Comparison of Average Overlapping Area (OA) for individual rule guidance between diffusion models trained on pixel and latent space. OA is computed using the full dataset as reference.

G.3. Composite Rule Guidance

In the task of composite rule guidance, the allocation of suitable weights to each rule is crucial for effective rule-based guidance. Table 14 shows the performance associated with various weight assignments. Generally, we observed that amplifying the weight assigned to a specific rule tends to decrease the loss pertinent to that rule. However, excessively concentrating the weight on a single rule can lead to a deterioration in performance, as evidenced by the configuration with a 40-1-4 weight assignment with an overly heavy emphasis on chord progression (CP).

Additionally, we investigated the impact of the sample count n on composite rule guidance, as shown in Table 15. The observed trend is consistent with that in individual rule guidance: using a greater number of samples at each step results in a lower loss. Another noteworthy observation is that combining SCG with other guidance methods (e.g., classifier guidance) and using a smaller sample count n (such as 4) can yield better outcomes than using SCG alone with $n = 16$. This improvement occurs because classifier guidance provides a coarse guidance signal, making it easier to identify advantageous directions based on these preliminary signals. As expected, the hybrid approach with a larger sample count $n = 16$ achieves the lowest loss. Remarkably, the losses in this case are similar to those in individual rule guidance, despite being achieved simultaneously.

Weight	PH ↓	ND ↓	CP ↓	OA full ↑
40-1-1	0.004 ± 0.005	0.218 ± 0.243	0.447 ± 0.226	0.901 ± 0.003
40-1-2	0.004 ± 0.004	$\mathbf{0.215 \pm 0.193}$	$\mathbf{0.392 \pm 0.206}$	$\mathbf{0.905 \pm 0.007}$
40-1-4	0.004 ± 0.004	0.251 ± 0.236	0.418 ± 0.216	0.884 ± 0.011
100-1-1	$\mathbf{0.003 \pm 0.002}$	0.236 ± 0.244	0.434 ± 0.229	0.903 ± 0.005

Table 14. Composite rule guidance using Classifier + SCG-4 with different weight on each rule. The weight column displays the weight in the order of PH, ND and CP.

Method	n	PH ↓	ND ↓	CP ↓	OA full ↑
SCG	16	0.014 ± 0.009	0.466 ± 0.648	0.446 ± 0.205	$\mathbf{0.909 \pm 0.005}$
DPS-NN + SCG	4	0.003 ± 0.004	0.392 ± 0.612	0.486 ± 0.270	0.826 ± 0.005
Classifier + SCG	4	0.004 ± 0.005	0.218 ± 0.243	0.447 ± 0.226	0.901 ± 0.003
DPS-NN + SCG	16	$\mathbf{0.002 \pm 0.007}$	0.238 ± 0.531	0.313 ± 0.231	0.844 ± 0.007
Classifier + SCG	16	0.003 ± 0.005	$\mathbf{0.148 \pm 0.203}$	$\mathbf{0.284 \pm 0.197}$	0.894 ± 0.007

Table 15. Effect of number of samples n on composite rule guidance.

H. Rule-Guided Generation Survey

H.1. Details

To evaluate the rule alignment of our approach SCG compared to two baseline methods, we designed a listening test. We extracted three musical rules (Pitch Histogram, Note Density, and Chord Progression) from various segments in our dataset. Next, we created music samples lasting 10.24 seconds each, adhering to these three rules. We produced four samples for each guiding method, including our own, resulting in a total of 12 samples.

We recruited 15 participants with substantial musical experience for our survey. We gathered information on their musical backgrounds, including the number of years they have been playing music, their years of formal music education, and the instruments they play. A significant portion of the participants have over 10 years of experience in both playing music and formal musical study, as shown in figures 6 and 7. Figure 8 displays a diverse range of instrument expertise among participants, with a notable prevalence of piano players, aligning with our model's focus on piano music. This diversity and level of experience make the participants well-suited for analyzing the music's rule alignment and musicality.
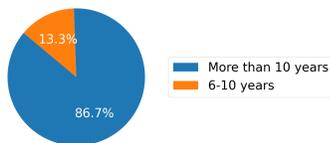

Figure 6. Years of playing music
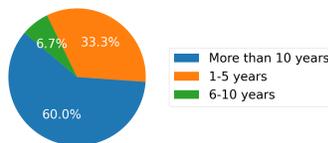
Figure 7. Years of formal music study
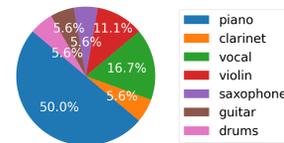
Figure 8. Instruments of participants

The dimension we evaluate about the sample quality are rule alignment, creativity, coherence and overall rating. Question 1-3 are about rule alignment, which evaluates the performance of guidance. Creativity refers to whether samples are musically interesting or not. For example, if one segment is static, then such sample is not creative. Coherence refers to whether the samples align with basic musical common knowledge. For instance, if one segment contains many random notes, then such sample sounds chaotic and it is not coherent to human's sense of good music. The overall rating is evaluated by participants, where they give a score solely based on their preference to the samples.

Each evaluation dimension is rated on a scale up to 5 points. For rule alignment, a Likert scale is used, where "completely unaligned" is rated as 1 and "perfectly aligned" as 5. The average score from questions 1-3 determines the rule alignment score. Creativity is initially scored out of 3 points, which is then normalized to a 5-point scale. The average of questions 4 and 5 calculates the creativity score. In question 4, "No" scores 1 point, "Maybe" 2 points, and "Yes" 3 points. For question 5, both "Too Simple" and "Too Complex" score 1 point, while "Moderate" scores 3 points.

For coherence, the maximum score is 4 points, later normalized to 5 points. The average from questions 6-8 gives the

coherence score. In question 6, "Many" errors score 1 point, "Some" 2 points, "A Few" 3 points, and "None" 4 points. In question 7, "Mainly incoherent harmonic motion" scores 1 point, "Mainly incoherent harmonic motion with some coherence" 2 points, "Mainly coherent harmonic motion with some incoherence" 3 points, and "Coherent harmonic motion" 4 points. Question 8 uses a tailored scoring system to fit the 4-point scale: "Poor" is valued at 4/3 points, "Moderate" at 8/3 points, and "Highly Engaging" at 4 points.

The overall rating also utilizes a Likert scale, with "Poor" equating to 1 point, "Fair" 2 points, "Good" 3 points, "Very Good" 4 points, and "Excellent" 5 points.

Regarding the music rules (Pitch Histogram, Note Density, and Chord Progression), we generated the entire sample conditioned on pitch histogram. To assess note density and chord progression, the music segment is divided into 8 equal-length segments or windows. We then analyze and compute the note density and chord progression within each of these windows. The survey consists 9 questions in total, investigating SCG's rule alignment and sample's musicality.

H.2. Survey

The first three questions of the survey are studying the rule alignment of guidance mechanisms. The answer of these three questions are all classified into 5 categories instead of binary choices because rule alignment can be effective for parts of the music sample. For example, given a 10 second music sample, the first 5 seconds of the sample has the perfect alignment, and the last 5 seconds of the sample does not align with provided rules at all. In this case, only binary classification on how effective the guidance is would not be enough to distinguish such sample. Thus, we construct 5 options instead of binary choices.

Question 1: On a scale of 1 to 5, how well does the pitch histogram in the sample music match the provided histogram? (1 indicating the least alignment, with 5 indicating the most alignment)

The options are:

- Completely unaligned (1): The pitch histogram in the sample music is completely different from the provided pitch histogram.

- Somewhat unaligned (2): The pitch histogram in the sample music is somewhat different from the provided pitch histogram, with a small portion of the segment aligned.

- Moderately aligned (3): The pitch histogram in the sample music is somewhat aligned with the provided pitch histogram, with a small portion of the segment not aligned.

- Mostly aligned (4): The pitch histogram in the sample music is mostly aligned with the provided pitch histogram.

- Perfectly aligned (5): The pitch histogram in the sample music is perfectly aligned with the pitch histogram.

Besides the generated, we show the participants the image of pitch histogram that are used for guidance. Note that Pitch histogram is the distribution of notes. Question 1 focuses on the alignment of pitch histogram, which means whether distribution of notes in the given sample follows the given pitch histogram. The question directly evaluates how effective the conditioning on pitch histogram is.

Question 2: On a scale of 1 to 5, how would you rate the alignment of the note density of the sample music compared to the note density provided in the above youtube video? (1 being the least aligned, 5 being the most aligned, take a look at the piano roll image would be a good idea)

The options are:

- Completely unaligned (1): The note density in the sample music significantly differs from the provided music segment, leading to a large disparity in musical texture and pacing.

- Somewhat unaligned (2): The note density in the sample music somewhat differs from the provided music segment, causing a noticeable disparity in musical texture and pacing.

- Moderately aligned (3): The note density in the sample music is somewhat aligned with the provided note density, but with a perceptible mismatch in musical flow and rhythm.

- Mostly aligned (4): The note density of the sample music closely matches the provided note density, with slight differences in how notes are spaced and arranged.

- Perfectly aligned (5): The note density of the sample music aligns perfectly with the provided note density, reflecting a very similar density pattern in the distribution of notes.

Question 2 evaluates the alignment of note density. Note density refers to the frequency and distribution of musical notes in a piece, indicating how many notes occur over a specific time or within a certain section of the music. In other words, note density reflects the texture and pacing in music segments. Thus, under a successful guidance of such rule, the texture and pacing of generated samples would be similar to the density pattern of corresponding segments in the distribution of notes.

Question 3: On a scale from 1 to 5, how well does the chord progression in the sample music match the provided chord progression, focusing on their functional harmony and general sequence rather than specific chord inversions. (1 indicates minimal alignment, with pronounced differences in chord progression, 5 signifies complete alignment, with the chord progressions being very similar or identical)

The options are:

- (Completely Unaligned): The bass line of the chord progression in the sample music significantly deviates from the provided progression, resulting in a clear disparity in harmonic structure and musical direction.

- (Somewhat Unaligned): Observable differences in the chord progression between the sample music and the provided example lead to a discordant sound and a disrupted musical flow.

- (Moderately Aligned): The chord progression in the sample music is somewhat consistent with the provided progression, with only minor discrepancies in the sequence or harmony.

- (Mostly Aligned): The chord progression in the sample closely mirrors the provided progression, with only negligible variations that don't substantially affect the overall harmonic continuity.

- (Perfectly Aligned): The chord progression in the sample music perfectly matches the provided progression, ensuring a cohesive and harmonious harmonic structure throughout.

Chord progression guides the music segment sounding more reasonable. Such questions asks about the alignment of chords, where effectively evaluates the controllable generation based on given chord progression.

The subsequent five questions explore the musicality of the generated samples, encompassing both creativity and coherence. The primary aim of rule-based guidance is to enhance the auditory appeal of the samples, making them more enjoyable to listeners. A generation is not considered successful if it fails to be aesthetically pleasing, regardless of achieving perfect alignment with all three specified rules.

Questions 4 and 5 focus on evaluating the creativity of the music sample.

Question 4: Do you like the music based on your personal taste?

The options are:

- Yes

- No

- Maybe

Questions 4 directly asks whether the participants like the music based on their personal taste. The evaluation from listeners with substantial musical experience illustrates the quality of model generation.

Question 5: What do you think of the complexity of this music?

The options are:

- Too Simple

- Moderate

- Too Complex

Question 5 assesses the complexity of the music, indicating that a moderate level of complexity is optimal. Music that is either too simple or too complex is considered to detract from the quality of the sample.

Questions 6 to 9 are designed to assess the coherence of the music sample.

Question 6: How many elements in the sample that seem out of place or random?

The options are:

- None

- A Few

- Some

- Many

Question 6 aims to evaluate on the generation quality of the model. Because the sample is composed by the model instead of human, such sample might have random notes. The random elements would break the entity of the music segment and the pleasure of listening for participants.

Question 7: How coherent do you find the harmony in the excerpt to be?

The options are:

- Coherent harmonic motion

- Mainly coherent harmonic motion with some incoherence

- Mainly incoherent harmonic motion with some coherence

- Mainly incoherent harmonic motion

Question 7 assesses the harmonic coherence of the generated sample, specifically evaluating if the music segment appears harmonically random or structured.

Question 8: How would you rate the appropriateness and engagement of the texture in the sample music, considering the layers and how they combine?

The options are:

- Highly Engaging

- Moderate

- Poor

Question 8 examines the texture of the sample music, focusing on whether the music presents an overly complex or disordered structure.

Question 9 solicits the overall rating of the music, where participants rate the sample according to their personal preferences.

Question 9: Overall Rating: On a scale of 1 to 5, how would you rate this sample? (1 being lowest, 5 highest)

The options are:

- Poor: The music lacks appeal in many aspects and does not engage the listener.

- Fair: The music has some redeeming qualities but falls short in several areas.

- Good: The music is enjoyable and well-composed, though it may have a few minor flaws.

- Very Good: The music is engaging and impressive, showing high levels of creativity and skill.

- Excellent: The music is outstanding in all respects, offering a deeply satisfying and memorable listening experience.