Controllable Music Synthesis with Diffusion Probabilistic Models

Bryan Chiang
Stanford University
bhchiang@stanford.edu

Natasha Ong
Stanford University
natashao@stanford.edu

Code: https://github.com/bryanhpchiang/symbolic-music-diffusion

Abstract

Conditional music synthesis is a difficult task due music's abstract nature, varying structure, and lack of task-specific data available. In this work, we propose to take an editing approach to controllable music generation to synthesize realistic samples from user-provided edits of symbolic music. We leverage recent advances in diffusion modelling that enable task-agnostic post-hoc conditioning on arbitrary user input at test time. Our diffusion model trained over a subset of the Lakh MIDI dataset outperforms our autoregressive baseline for unconditional music generation. Based on quantitative and qualitative evaluations, we found that our conditional generation experiments were able to generate samples that successfully incorporated user inputs on composition and synthesis tasks.

1 Introduction

In recent years variational autoencoders (VAEs) and generative adversarial networks (GANs) have been a popular and successful way of representing discrete and continuous domains, producing high-quality samples in domains such as art and music. However, GANs suffer from unstable training and less diverse generation due to the adversarial training of the generator and discriminator and VAEs rely on a surrogate loss (ELBO). As such, researchers have explored using denoising diffusion probabilistic models (DDPMs)(1)(2), which learns to invert a diffusion process from data to Gaussian noise, and have found similar success(3; 4; 5).

DDPMs sample through an iterative refinement process, thereby enabling post-hoc conditioning of models trained unconditionally. Though traditional iterative refinements operate on continuous data and thus struggle with discrete data like symbolic music, training DDPMs on the continuous latents of low-level VAEs has been found to overcome these limitations (5). As such, VAEs coupled with DDPMs enable various editing-based creative applications, where user edits are proposed at test time after our model has already been trained.

1.1 Problem Statement

To our best knowledge, there does not exist an editing approach to controllable music generation. More specifically, rather than synthesizing a new example from scratch, this procedure allows the user to provide a rough outline or sketch of the desired musical sample as input into a pre-trained generator, which then synthesizes the sketch in the style of the training data. Having such an approach, like SDEdit(4) for art, encourages creative and artistic expression in music composition, a field that historically requires a lot of time and dedication to understand and explore.

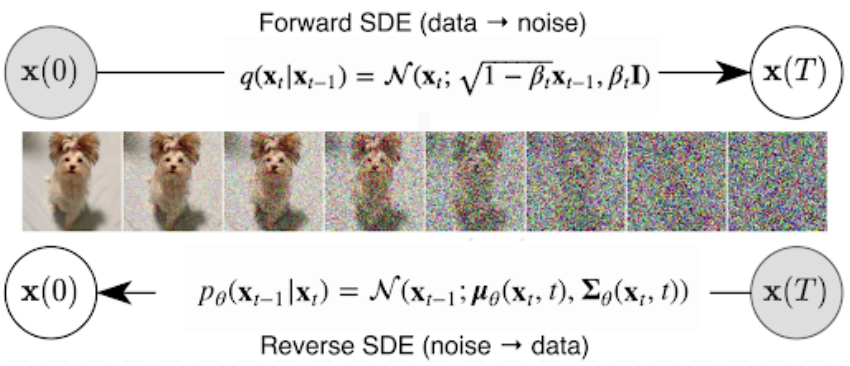

Figure 1: The forward and reverse diffusion processes. The left is the original data sample, and the right is an isotropic Gaussian distribution.

2 Related Work

2.1 Diffusion Models

Score-based generative models, such as denoising diffusion probabilistic models (DDPMs) (1) and score matching with Langevin dynamics (SMLD) (6), are probabilistic methods that sequentially corrupt data with gradually increasing amounts of noise (forward process), and then learn a generative model to retrieve the original data (reverse process). We can view such models as reversible SDEs that govern the diffusion of a data point into random noise. Since the conditional reverse-time SDE can be efficiently estimated from unconditional scores, enabling new applications with information not available during the training process without any retraining (7). In this vein, SDEdit (4) approaches image editing and synthesis through the use of generative modeling with stochastic differential equations (SDEs). They look at forward and reverse processes: in the forward process, the SDE progressively adds Gaussian noise as time elapses. They leverage the reverse stochastic process such that the model can take a noisy input and denoise to ultimately produce an image of high quality. They found that their model outperforms other GAN-based methods on stroke-based image generation, such as StyleGAN2-ADA (8) and in-domain GAN (9), and also achieves similar performances on stroke-based image editing that have no or minimal modifications.

2.2 Audio Synthesis

Building on top of score-based generative models, DiffWave (3) proposes a diffusion model for both conditional and unconditional generation of waveforms, achieving high-fidelity audio on a variety of waveform generation tasks with performance comparable to existing state-of-the-art methods. Similarly, (5) learn to perform unconditional generation of symbolic music through diffusion models by first converting to continous embeddings through a pre-trained VAE encoder. Both models were empirically demonstrated to outperform autoregressive and GAN baselines. In this work, we work with discrete music provided in the MIDI format as in (5) instead of continuous audio waveforms. We aim to investigate whether the success of diffusion-based editing techniques, namely SDEdit (4), can also be applied in the audio domain.

3 Technical Approach

3.1 Diffusion Models

Diffusion models consist of both a forward process and reverse process, illustrated in Figure (1).

In the forward process, we corrupt a given a data point sampled from the real data distribution $x_0 \sim q(x)$ by adding small amounts of Gaussian noise over T steps, given by Equation (1). The noise variance schedule is given by $\{\beta_t \in (0, 1)\}_{t=1}^{T}$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

For an entire sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we have $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$ by Chain Rule. (1) found that by the reparametization trick, any noisy sample at time step t can be represented in closed form by $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t|\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$. $\alpha_t = 1-\beta_t$ and $\hat{\alpha}_t = \prod_{i=1}^{t} \alpha_t$.

In the reverse process, we aim to recreate a true sample from an input sampled from the standard Gaussian as $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We aim to learn a model $p_\theta(\mathbf{x})$ to compute the reverse conditional probabilities $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ since they are intractable to compute. The reverse process is given in Equation 2.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2}$$

In practice, the variance term is set to a untrained time-dependent noise schedule, (1) found $\mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) = \tilde{\beta}_t = \frac{1-\hat{\alpha}_{t-1}}{1-\hat{\alpha}_t}\beta_t$ to work well. To train the model μ_θ, the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$, giving us the true mean $\tilde{\mu}_t$ in Equation (3).

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\mathbf{z}_t) \tag{3}$$

z_t is noise term sampled from the standard normal. We find the variational lower bound to optimize the negative log-likelihood, which minimizes difference between the predicted mean μ_θ and $\tilde{\mu}$. Since $\mathbf{x}_t$ is known, the reparameterize the model to predict the noise term as $\mathbf{z}_\theta$ instead. Our final L2 loss term in Equation (4) from (1), which was found to yield higher-quality samples, trains our learned diffusion model with pairs of (x_t, x_0).

$$L(\theta) = \mathbb{E}_{x_0,z}[||\mathbf{z}_t - \mathbf{z}_\theta(\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{z}, t)||^2] \tag{4}$$

x_0 is the embedding sequence of the original symbolic audio, x_t represents the forward process at time step t. t is sampled uniformly between timesteps 1 and T, $\epsilon \sim \mathcal{N}(0, I)$, and $\mathbf{z}_\theta$ is our learned diffusion model. Starting with the noise schedule $\beta_1, \beta_2, \ldots, \beta_n$, we have $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\alpha_t = 1 - \beta_t$. Here, the diffusion model, given by $\mathbf{z}_\theta(\mathbf{x}_t, \sqrt{\bar{\alpha}})$, is implemented with a transformer architecture (10) following the methodology from (5), illustrated in Figure (3). For the transformer architecture, we use $L = 6$ encoder layers with $H = 8$ self-attention heads and a residual fully-connected layer. A 128-dimensional sinusoidal positional encoding is appended to the inputs to the encoder layers to capture the temporal context of the latent embeddings.

The full procedures for training and unconditional sampling are outlined in 2.

3.2 Controllable Synthesis

Given a MIDI note sequence s', the corresponding latent representation is denoted as $s \in \mathbb{R}^{1\times k\times 42}$, where k is the number of latents determined by the length of the note sequence and 42 is the dimensionality of the latent space.

Given a pre-trained diffusion model and edited sample s, the full conditional generation process is detailed in Algorithm (1). Generation is only applied for the unmasked inputs, masked regions of the edited sample are fixed and should not be touched by the generation process. The mask is set to $m = 1$ for fixed regions, and $m = 0$ for the regions to generate. The T used for conditional generation is typically lower than the T used for training to avoid completely obscuring the provided user input with Gaussian noise.

We consider two types of controllable synthesis tasks for a user-provided note sequence u', corresponding latent u. For composition within the output MIDI space, the edited sample s' is created by inserting u' within a reference note sequence u' starting from time t seconds, overwriting the

Algorithm 1 Reverse process for music synthesis from edited samples.

Require: mask $\mathbf{m}$, edited sample $\mathbf{s}$, T steps, noise schedule $\beta_1, \ldots, \beta_T$

$\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_T = \sqrt{\alpha_T}\mathbf{s} + \sqrt{1-\alpha_T}\mathbf{z}_0$ ▷ Edited input initialize with noise

for $t = T, \ldots, 1$ **do**

$\mathbf{z}_1, \mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z}_1 = \mathbf{z}_2 = 0$

$y = \sqrt{\bar{\alpha}_t}s + \sqrt{1-\hat{\alpha}_t}\mathbf{z}_1$ if $t > 1$, else $\mathbf{z}_1$ ▷ For fixed regions

$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\mathbf{z}_\theta}(\mathbf{x}_t, \sqrt{\bar{\alpha}_t})) + \sigma_t \mathbf{z}_2$ ▷ Reverse denoising step

$\mathbf{x}_{t-1} = \mathbf{x}_{t-1} \odot (1 - \mathbf{m}) + \mathbf{y} \odot \mathbf{m}$

end for

Algorithm 1 Training

Input: $q(x_0)$, N steps, noise schedule $\beta_1, ..., \beta_N$

repeat

$x_0 \sim q(x_0)$

$t \sim \mathbb{U}(\{1, ..., N\})$

$\sqrt{\bar{\alpha}} \sim \mathbb{U}(\sqrt{\bar{\alpha}_{t-1}}, \sqrt{\bar{\alpha}_t})$

$\epsilon \sim \mathcal{N}(0, I)$

Take gradient descent step on

$\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \sqrt{\bar{\alpha}}) \right\|^2$

until converged

Algorithm 2 Sampling

Input: N steps, noise schedule $\beta_1, ..., \beta_N$

$x_N \sim \mathcal{N}(0, I)$

for $t = N, ..., 1$ **do**

$\epsilon \sim \mathcal{N}(0, I)$ if $t > 1$, else $\epsilon = 0$

$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, \sqrt{\bar{\alpha}_t}) \right) + \sigma_t \epsilon$

end for

return x_0

Figure 2: General algorithm for training and unconditional from (5).

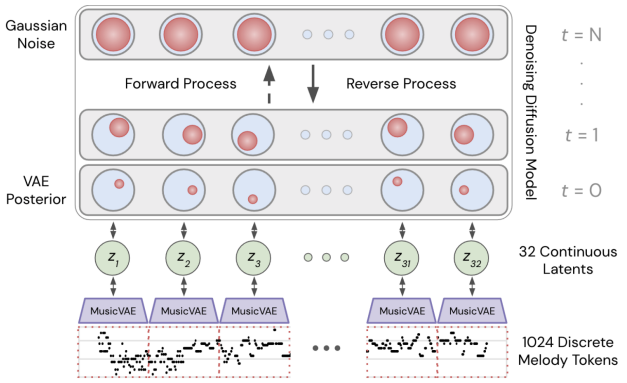

Figure 3: Our model architecture/framework

existing notes. We then convert s' to the latent representation s for generation. For composition within the latent space, the reference and user-provided note sequences are both converted into their latent representations to obtain s and r, respectively. u is then inserted into r starting from latent number l, overwriting the existing latents. For synthesis, we have no reference melody and directly set the edited samples to $s = u$.

4 Experiments

4.1 Dataset

We use the Lakh MIDI Dataset (LMD) (11), which contains over 170,000 MIDI files of various short melodies between 30 seconds to a few minutes long. We only use a subset, namely 500 MIDI files from this dataset due to the long pre-processing time. Since our data is discrete note sequences in the MIDI format, we pre-trained 2-bar melody MusicVAE model (12) to embed the notes into the VAE's continous latent space. Training and evaluation took place on 32 count context windows of latents representing musical phrases of 64 bars (1024 tokens) each.

4.2 Editing Tasks

Our problem and its corresponding tasks and experiments are three-fold. First, we aim to reproduce the results from (5) by training the diffusion model on the Lakh MIDI dataset (11). Second, we build on this reproduced diffusion model by exploring two types of input edits: composition and synthesis. Third, we explore making these edits it two different spaces: output space and latent space.

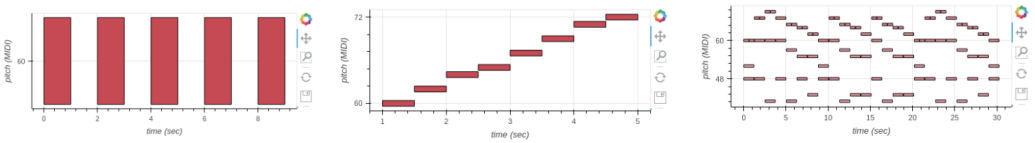

Figure 4: MIDI of 3 Melodies (L to R): Repeated Middle C, C-Major Scale, Twinkle Twinkle Little Star. x-axis: time, y-axis: MIDI pitch

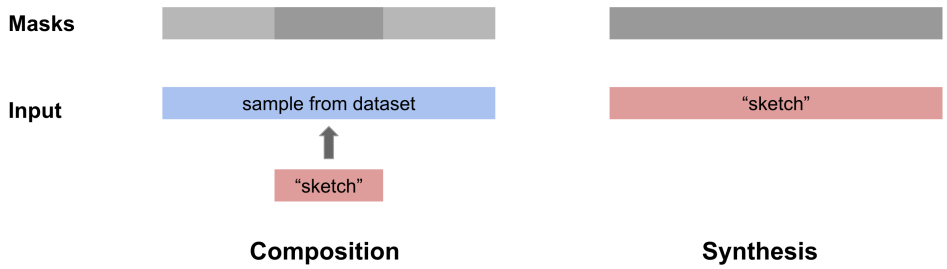

Figure 5: We explore two types of user input edits: composition and synthesis. Red indicates the user-provided note sequence input, blue is the longer, reference note sequence chosen from the dataset. The darker parts of the mask indicate which parts of the edited sample should be touched by the conditional generation process.

We benchmark the performance of these editing methods using 3 melodies4 of increasing complexity: **repeated Middle C**, **C-Major Scale**, and **Twinkle Twinkle Little Star**.

4.3 User Input Edits

We explore two types of input edits5: composition, where a user-generated melody is embedded into a longer reference melody, and synthesis, where no reference melody is used. Masks indicate which areas should be generated.

4.4 Editing Space

We experiment with performing the edits in both the output space (discrete MIDI files) and the latent space (continuous embeddings).

For both cases, we will follow the method from SDEdit (4) to recover a denoised sample from the given user input in the latent space by applying the reverse process of the diffusion model. We expect to see that the denoised samples will incorporate the simple user-guided input to create richer and more complex melodies according to our evaluation below. In addition, we expect to find correlations between specific dimensions in our latent embeddings and resulting metrics. For fairness, we use the same architecture as the diffusion model and perform training and evaluation over the same dataset.

4.5 Baseline

For our project we have two baselines, one for our first task, which was to benchmark the performance of our DDPM model on unconditional audio synthesis. More specifically, we compare our DDPM model (prior to altering to make it work with user edits) against TransformerMDN (baseline model). TransformerMDN is an autoregressive transformer with a mixture density(13) output layer with 100 mixture Gaussians to provide sufficient mode coverage.

Our second baseline for conditional generation is used for our main experiments to help us compare and interpret our generated samples and results. More specifically, use the raw edited samples as our baseline against the generated results. This enables us to see how well the reverse diffusion process integrates the provided user input.

5 Results

5.1 Evaluation

Qualitatively, we plan on incorporating human evaluation since music is subjective on many levels. We will listen to the various generated audio for the different types of edits (composition vs synthesis) and editing space (output vs latent), taking note of the perceived coherence of the generated audio as well as the influence/impact of the input sketch on these generated samples. Moreover, we will look at the MIDI transcription (images of notes and durations). Comparing the transcriptions of the unedited audio, edited audio, and generated/sampled audio, we will judge whether the user edits seem to have been incorporated (either in style, notes, note lengths, etc) into our samples.

Quantitatively, we use metrics that broadly measure framewise self-similarity and latent space distributions.

5.1.1 Framewise Self-similarity

Since coherent music tends to be locally similar in many ways, we capture this concept by evaluating our models with a modified Overlapping Area (OA) metric. Inspired by (5), we consider a sliding window of 4 measures and a hop size of 2 measures to capture local pitch and duration statistics across the piece.

For each of the 4 measure windows, we compute the mean and variance of pitch and duration, where the former captures melodic similarity and the latter captures rhythmic similarity. These values describe a Gaussian PDF for pitch and duration for each frame, which we can use to compute the Overlapping Area (OA)(14) of two adjacent frames $(k, k+1)$. We define the statistics for frame $k, k+1$ as $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$ respectively. The overlapping area is then:

$$OA(k, k+1) = 1 - \text{erf}\left(\frac{c - \mu_1}{\sqrt{2}\sigma_1^2}\right) + \text{erf}\left(\frac{c - \mu_2}{\sqrt{2}\sigma_2^2}\right)$$

, where c represents the intersection point between Gaussian PDFs with $\mu_1 < \mu_2$ and erf represents the Gauss error function.

Using the mean (μ_{OA}) and variance (σ_{OA}) of the OA aggregated over all adjacent frames, we infer the Consistency and Variance for a set of MIDI samples. We then compute the normalized relative similarity of pitch and duration consistency and variance to the training set (μ_{GT}, σ_{GT}):

$$\text{Variance} = \max\left(0, 1 - \frac{|\sigma_{OA}^2 - \sigma_{GT}^2|}{\sigma_{GT}^2}\right)$$

$$\text{Consistency} = \max\left(0, 1 - \frac{|\mu_{OA} - \mu_{GT}|}{\mu_{GT}}\right)$$

Values are clipped to 0 so samples with μ_{OA}, σ_{OA}^2 having a percent error greater than 100% from the ground truth are assigned zero relative similarity to the training set.

5.1.2 Latent Space Distributions

This metric measures the similarity of latent embeddings generated by the model. Though it does not measure long-term temporal consistency or quality of produced sequences, this metric does allow for comparisons between models. Specifically, we are measuring and evaluating the quality of the *intermediate* continuous representation before passing into the MusicVAE decoder. To measure the similarity of the latent embeddings, we use the Frechet distance (FD) (15) and Maximum Mean Discrepancy (MMD) (16) with a polynomial kernel, which measure the distance between the models' continuous output distributions and the original data distribution in latent space.

The Frechet distance between $\mathcal{N}(\mu_G, \Sigma_G^2)$ and $\mathcal{N}(\mu_R, \Sigma_R^2)$ for generated and real samples respectively, is given by:

$$\text{FD}^2((\mu_G, \Sigma_G), (\mu_R, \Sigma_R)) = ||\mu_G - \mu_R||_2^2 + \text{Tr}\left(\Sigma_G + \Sigma_R - 2(\Sigma_G \Sigma_R)^{1/2}\right)$$

Meanwhile we calculate MMD using two kernels, radial basis function (RBF) and polynomial. Using a kernel k, the MMD for real (R) and generated (G) is given by:

$$\begin{aligned}\text{MMD}^2(R,G) &= ||\mu_R - \mu_G||_{\mathcal{F}}^2 \\ &= \mathop{\mathbb{E}}_{\mathcal{X}\sim R}[k(x,x')] + \mathop{\mathbb{E}}_{\mathcal{Y}\sim G}[k(y,y')] - 2\mathop{\mathbb{E}}_{\mathcal{X},\mathcal{Y}\sim R,G}[k(x,y)]\end{aligned}$$

5.2 Unconditional Generation

We trained and evaluated our autoregressive model (TransformerMDN) and score-based model (DDPM) on our data subset directly in the latent space over the embeddings. To evaluate the quantitative performance of the TransformerMDN and diffusion model, we compared the framewise self-similarity (Table 1) and latent-space distribution metrics (Table 2).

	Pitch C	Pitch Var	Duration C	Duration Var
Diffusion	0.93	0.90	0.93	0.98
TransformerMDN	0.97	0.84	0.99	0.71

Table 1: Self-similarity (coherence) evaluation of unconditional samples. `Var` indicates variance, and `C` indicates consistency. Higher values are better, capped at 1.

	FD	MMD rbf	MMD poly.
Diffusion	1.53	0.33	0.94
TransformerMDN	1.58	0.36	1.03

Table 2: Latent space evaluation of unconditional samples. Lower values are better.

Our results, albeit trained on a smaller dataset, confirm the results of (5). Overall, the diffusion model outperforms the baseline TransformerMDN model. In the output space, we hypothesize that the lower variance values for the autoregressive baseline compared to the diffusion is because the baseline is unable to simultaneously incorporate information from all latents, which is key for capturing rhythmic dependencies across time. In the latent space, we see that although the autoregressive baseline starts with a mixture of Gaussian priors at its output layer, it is still outperformed by the distribution learned from scratch by the diffusion model from scratch. This may be due to the teacher forcing leading to exposure bias and divergence from the data distribution for the autoregressive baseline. The discrepancies between our values and the values reported in (5) are likely because of our usage of small subset of data. We chose to use a smaller dataset at the expense of worse unconditional generation metrics since we wanted to focus the bulk of our efforts on our conditional generation experiments and exploring various configurations for controllable synthesis.

5.3 Conditional Generation

Qualitative Results: Overall, from examining the MIDI transcriptions (images) for edits in the latent and output space, we notice that latent space editing tends to change the original reference audio in a stylistically coherent manner. On the other hand, editing in the output space transfers the user edits more directly into the generated samples, thus not embedding in a "smoother" manner with the rest of the reference audio. We can see this when looking at the transcriptions of generated samples for the repeated C melody in Figure 6 and Figure 7. Changes are shown in red.

For the latent space edits, we notice that the generated audio transfers the repeated style/nature of our user "sketch". These reference melody contains notes that are repeated twice, which these changes mimic as well. Moreover, we see that the middle C note is not preserved in the changed area and

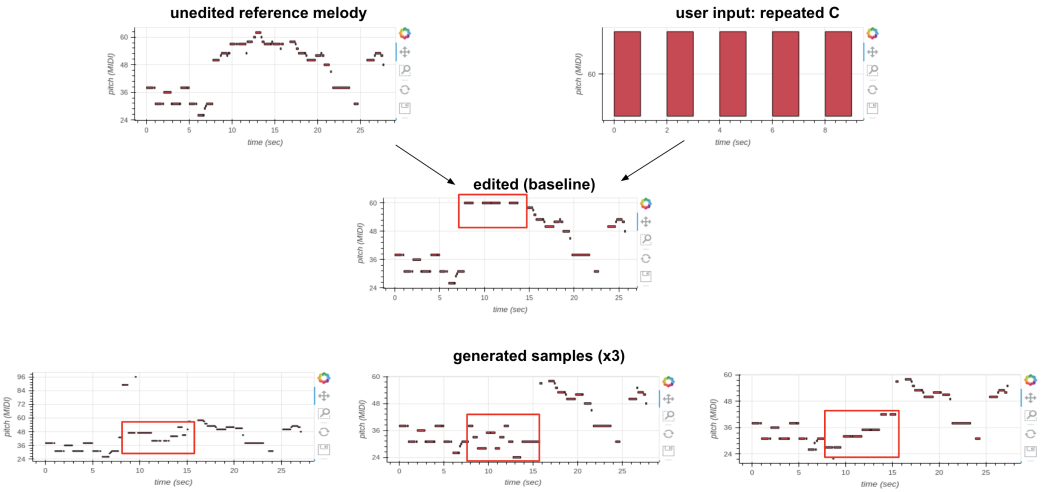

Figure 6: Repeated C melody with composition and latent space editing

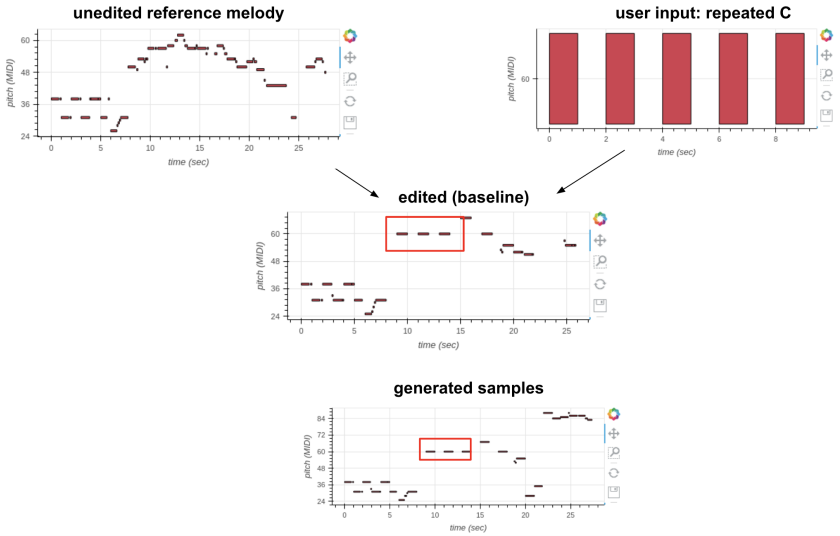

Figure 7: Repeated C melody with composition and output space editing

instead consist of notes that have already appeared in the audio, thus seeming to be more coherent with less abrupt rhythmic and note changes. Listening to the actual generated audios confirmed these findings as it was difficult to discern exactly where the changes were made without re-referencing the transcription images.

For the output space edits, the generated samples tended to contain the repeated c notes directly without much change. The note pitch (ie middle C) persisted in the samples as well as the note lengths, as seen clearly when comparing the editing transcription and generated samples in Figure 7. These observations were reinforced when listening to the generated audio as the middle C-notes stood out very distinctively as the notes did not change and were repeated many times. Thus, it was very easy to tell exactly where these edits were made and felt more like the two audios were direct spliced together instead of the user edit being smoothly embedded into the reference audio.

These findings were consistent for the repeated C and C-major scale melodies. However, listening to Twinkle Twinkle Little Star and the original reference melody, we realized that the lengths of these two samples are almost the same. Thus, when listening to the latent space edits, the "changes" represented most of the notes in the samples. Though aspects of the melody seemed to be incorporated

in the samples, most of the original reference melody was lost. The same held true for samples produced by edits in the output space.

For synthesis, we realize that since there are no reference comparison audios, evaluation was harder in terms of not knowing what really sounded "good" or particularly coherent. We realize, however, that this may also be due to the original MIDI files in the dataset not being very coherent or "music-like" in the first place.

Quantitative Results: We evaluated our 3 melodies using composition and synthesis editing types, with edits being made in latent and output space for composition. These combinations were evaluated using the framewise self-similarity and latent space distribution metrics from Section 5.1.1 and 5.1.2, respectively, and can be referenced in Table 3.

We notice that for Repeated C and C-Major Scale, latent-space editing generally outperforms output-space editing for our generated samples across all metrics. Thus, quantitatively, we can conclude that for these two melodies, editing in the latent space for composition produces samples that are more coherent and similar in distribution to the original data.

However, this observation does not hold for the more complicated Twinkle-Twinkle Little Star melody, where we see that editing in the output-space seems to perform better under self-similarity metrics while latent-space seems to perform better under latent-space distribution metrics.

Nevertheless, intuitively, it makes sense that editing in the output space would produce worse results for metrics that measure distance between latent-space distributions as it is unclear how editing in the output space translates to the latents. As for why the output-space edits outperform latent-space edits, we hypothesize that the length of the user sketch vs reference melody has some impact. Additionally, as the complexity increases perhaps our latents do not adequately capture the various musical aspects of a melody. More specifically, this melody is much longer in time and contains ascending and descending notes that cover a larger contextual area and are thus harder to "learn"/capture, as visualized in 4. On the other hand, editing in the output space directly inserts this melody, and may potentially overcome this challenge. However, more experiments are required to test this hypothesis.

For Synthesis, we did not evaluate using latent space distribution metrics as these do not make much sense as it is unclear what to compare the distribution against. For our self-similarity metrics, our generated samples, quantitatively, are really good as the values are close to the maximum ("ideal") value of 1 across all 3 melodies. Intuitively, we may be to explain these results by noting that synthesis only contains one input while composition requires the user input to be "embedded" into a longer and oftentimes different melody. More specifically, composition may be a more difficult task as the two melodies may have very different characteristics in mood, rhythm, tempo, pitch, etc that may make it more challenging to merge into something that is coherent.

6 Conclusion

Conditional synthesis of music is difficult due to music being abstract and requiring more task-specific expertise compared to images. In our project, we proposed to take an editing approach to controllable music generation based on recent advances in diffusion models to synthesize realistic samples from user-provided edits of symbolic music.

For composition-type edits, editing in the latent space outperformed editing in the output space for the repeated C and C-Major scale melodies. Moreover, edits in the latent space tended to more smoothly and coherently embed the user edit into the longer reference melody. However, the output space edits quantitatively performed better than the latent space edit for Twinkle Twinkle Little Star. For synthesis-type edits, our quantitative values were close to ideal, though limitations in the quality of audio samples in the dataset and lack of reference audios made it difficult to evaluate.

For future work, there are several ways to improve or iterate on our work. Due to limitations in time and scope, we were only able to run a handful of experiments and train on a smaller dataset.

Melody	Editing Description	Edited/Generated	Self-Similarity				Latent Distribution		
			Pitch		Duration			MMD	
			Var	C	Var	C	FD	poly	rbf
Repeated C	Composition, Output	Edited	0.89	0.99	0.50	0.98	**5.51**	11.36	**0.28**
		Generated	**0.94**	**1.00**	**0.79**	**0.99**	5.57	**11.31**	0.30
	Composition, Latent	Edited	0.88	0.99	0.98	0.49	2.12	2.43	0.08
		Generated	**0.94**	**1.00**	**0.99**	**0.79**	**1.77**	**1.84**	**0.09**
	Synthesis	Edited	0.98	1.00	**0.98**	1.00	–	–	–
		Generated	**0.99**	1.00	0.97	1.00	–	–	–
C-Major Scale	Composition, Output	Edited	**0.98**	**0.99**	**0.50**	**0.97**	**3.35**	**4.33**	**0.27**
		Generated	0.49	0.97	0	0.96	4.54	5.00	0.30
	Composition, Latent	Edited	**0.91**	0.99	0.80	0.98	0.96	1.37	0.04
		Generated	0.88	0.99	**0.82**	**0.99**	**0.86**	**1.18**	0.04
	Synthesis	Edited	0.83	0.99	0.99	0.99	–	–	–
		Generated	**0.97**	**1.00**	0.99	**1.00**	–	–	–
Twinkle Twinkle	Composition, Output	Edited	0.71	0.96	**0.74**	**0.98**	10.10	13.6	**0.29**
		Generated	**0.93**	**0.98**	0.23	0.96	**9.44**	**11.6**	0.30
	Composition, Latent	Edited	0.75	0.96	**0.84**	**0.99**	**4.82**	**7.69**	**0.26**
		Generated	**0.81**	**0.97**	0.02	0.98	6.96	9.15	0.37
	Synthesis	Edited	**1.00**	1.00	1.00	1.00	–	–	–
		Generated	0.99	1.00	1.00	1.00	–	–	–

Table 3: Self-Similarity and Latent Space Distribution Metrics for Edited (baseline) and Generated Sample. Higher self-similarity values are better (more coherent) and can take on a maximum value of 1. Lower Latent Space Distribution values are better (lower distance between distributions)

Some ideas we have for future work as the following:

- Train our DDPM on a larger dataset. Due to compute resources and time constraints, we only trained our DDPM on 500 MIDI files. It would be interesting to explore training on a bigger dataset to see whether training this has recognizable differences in the quality of audio for our various edits.
- Experiment with editing locations in the output space. Since the mapping from MIDI space to latent, continuous space is unclear, we picked at arbitrary location to edit in the output space. Further work could investigate whether some sort of mapping between MIDI and latent space can be deduced. Moreover, it would be interesting to experiment whether there is an optimal location to edit in each MIDI file to produce better results.

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[2] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[3] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," 2021.

[4] C. Meng, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Image synthesis and editing with stochastic differential equations," *arXiv preprint arXiv:2108.01073*, 2021.

[5] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," 2021.

[6] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *arXiv preprint arXiv:1907.05600*, 2019.

[7] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[8] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," 2020.

[9] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," 2018.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

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

[12] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," 2019.

[13] C. M. Bishop, "Mixture density networks," 1994.

[14] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," 2020.

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[16] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012. [Online]. Available: http://jmlr.org/papers/v13/gretton12a.html