SYMBOlIC MUSIC GENERATION WITH DIFFUSION MODELS

Gautam Mittal1⋆ Jesse Engel2 Curtis Hawthorne2 Ian Simon2
1 University of California, Berkeley 2 Google Brain

gbm@berkeley.edu, {jesseengel,fjord,iansimon}@google.com

ABSTRACT

Score-based generative models and diffusion probabilistic models have been successful at generating high-quality samples in a variety of continuous domains. However, due to their Langevin-inspired sampling mechanisms, their application to discrete symbolic music data has been limited. In this work, we present a technique for training diffusion models on symbolic music data by parameterizing the discrete domain in the continuous latent space of a pre-trained variational autoencoder. Our method is non-autoregressive and learns to generate sequences of latent embeddings through the reverse process and offers parallel generation with a constant number of iterative refinement steps. We show strong unconditional generation and post-hoc conditional infilling results compared to autoregressive language models operating over the same continuous embeddings.

1. INTRODUCTION

Denoising diffusion probabilistic models (DDPMs) [1, 2] are a promising new class of generative models that can synthesize comparably high-quality samples by learning to invert a diffusion process from data to Gaussian noise. Unlike many existing deep generative models, DDPMs sample through an iterative refinement process inspired by Langevin dynamics [3], which enables post-hoc conditioning of models trained unconditionally [4–7] for creative applications.

Despite these exciting advances, DDPMs have not yet been applied to symbolic music generation because their iterative refinement sampling process is confined to continuous domains such as images [2] and audio [8,9]. Similarly, DDPMs cannot take advantage of the recent advances in modeling long-term structure [10–12] that use a two-stage process of modeling discrete tokens extracted by a separate low-level autoencoder.

In this paper, we demonstrate that it is possible to overcome these limitations by training DDPMs on the continuous latents of a low-level variational autoencoder (VAE) to generate long-form discrete symbolic music. Our key findings include:

- High-quality unconditional sampling of discrete melodic sequences (1024 tokens) with DDPMs through iterative refinement of lower-level VAE latents.
- DDPMs outperforming strong autoregressive baselines (TransformerMDN) in hierarchical modeling of continuous latents, partly due to a lack of teacher forcing and exposure bias during training.
- Post-hoc conditional infilling of melodic sequences for creative applications.

2. BACKGROUND

2.1 Denoising Diffusion Probabilistic Models

DDPMs [1, 2] are a class of generative models that define latents \(x_1, \ldots, x_N\) of the same dimensionality as the data \(x_0 \sim q(x_0)\). Diffusion models are comprised of a forward process and a reverse process. The forward process starts from the data \(x_0\) and iteratively adds Gaussian noise according to a fixed noise schedule for \(N\) diffusion steps:

\[
q(x_t|\frac{x_{t-1}}{\theta}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \quad (1)
\]

\[
q(x_{1:N}|x_0) = \prod_{t=1}^{N} q(x_t|x_{t-1}) \quad (2)
\]

where \(\beta_1, \beta_2, \ldots, \beta_N\) is a noise schedule that converts the data distribution \(x_0\) into latent \(x_N\). The choice of noise schedule has been shown to have important effects on sampling efficiency and quality [2,8].

The reverse process is defined by a Markov chain parameterized by \(\theta\) that iteratively refines latent point \(x_N \sim \mathcal{N}(0, I)\) into data point \(x_0\). The learned transition probabilities are defined as,

\[
p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \sigma_\theta(x_t,t)) \quad (3)
\]

\[
p_\theta(x_{0:N}) = p(x_N) \prod_{t=1}^{N} p(x_{t-1}|x_t) \quad (4)
\]

where the objective is to gradually denoise samples at each reverse diffusion step \(t\). In practice, \(\sigma_\theta\) is set to an untrained time-dependent constant based on the noise schedule, and [2] found \(\sigma_\theta(x_t,t) = \sigma_t = \frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t} \beta_t\) to have reasonable practical results, where \(\alpha_t = 1 - \beta_t\), and \(\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i\).
The training objective is to maximize the log likelihood of $p_\theta(x_0) = \int p_\theta(x_0, \ldots, x_N) dx_{1:N}$, but the intractability of this marginalization leads to the following evidence lower bound (ELBO):

$$
\begin{align*}
\mathbb{E}[\log p_\theta(x_0)] &\geq \mathbb{E}_q \left[ \log \frac{p_\theta(x_0, x_N)}{q(x_1:N|x_0)} \right] \\
&= \mathbb{E}_q \left[ \log p(x_N) + \sum_{t=1}^{N} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] \\
&= \mathbb{E}_q \left[ \log p(x_N) + \sum_{t=1}^{N} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right]
\end{align*}
$$

Additionally, [2] observed that the forward process can be computed for any step $t$ such that $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_0, (1 - \alpha_t) I)$, which can be viewed as a stochastic encoder. To simplify the above variational bound, [2] propose training on pairs of $(x_t, x_0)$ to learn to parameterize this process with a simple squared L2 loss. The following objective is simpler to train, resembles denoising score matching [5, 14] and was found to yield higher-quality samples:

$$
L(\theta) = \mathbb{E}_{x_0, \epsilon, t} \left[ \| \epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon, t) \|^2 \right]
$$

where $t$ is sampled uniformly between 1 and $N$, $\epsilon \sim \mathcal{N}(0, I)$, and $\epsilon_\theta$ is the learned diffusion model. [8] found that instead of conditioning on a discrete diffusion step $t$, it was beneficial to sample a continuous noise level $\sqrt{\alpha} \sim \mathcal{U}(\sqrt{\alpha_{t-1}}, \sqrt{\alpha_t})$ where $t \sim \mathcal{U}\{1, \ldots, N\}$ and $\alpha_0 = 1$.

We refer readers to Algorithms 2 and 3 based on [8] in our supplementary material\(^1\) for the full training and sampling procedure.

\(^1\)Supplementary material including additional implementation details, audio, and tables is available at https://goo.gl/magenta/symbolic-music-diffusion-examples

2.2 Variational Autoencoders

Variational autoencoders [15] are generative models that define $p(y, z) = p(y|z)p(z)$ where $z$ is a learned latent code for data point $y$. Additionally, the latent code is constrained such that $z$ is distributed according to prior $p(z)$ where the prior is usually an isotropic Gaussian. The VAE is comprised of an encoder $q_{\gamma}(z|y)$ which models the approximate posterior $p(z|y)$ and a decoder $p_\theta(y|z)$ which models the conditional distribution of data $y$ given latent code $z$.

The training objective is to maximize the log likelihood of $p_\theta(x) = \int p_\theta(y|z)p(z)dz$, but this marginalization is intractable, and we use the following variational bound maximized with $q_{\gamma}(z|y)$ as the approximate posterior:

$$
\mathbb{E}[\log p_\theta(y|z)] - KL(q_{\gamma}(z|y)||p(z)) \leq \log p(y)
$$

The flexible implementation of variational autoencoders allows them to learn representations over a wide variety of domains. Of particular interest to us are sequential autoencoders [13, 16] which use long short-term memory cells [17] to model temporal context in sequential data distributions.

In practice, there is a trade-off between the quality of reconstructions and the distance between the approximate posterior $q_{\gamma}(z|y)$ and the Gaussian prior $p(z)$. This makes sampling more difficult for VAEs with better reconstructions due to latent “holes” in the approximate posterior and is one of the primary shortcomings of these models.

3. MODEL

A diagram and description of our multi-stage diffusion model is shown in Figure 1. We refer readers to the sup-
plement for full implementation details.

3.1 Architecture

Our model learns to generate discrete sequences of notes (known as MIDI) by first training a VAE with parameters $\gamma$ on the sequences and then training a diffusion model to capture the temporal relationships among the $k$ VAE latents. Sequence VAEs such as MusicVAE are difficult to train on long sequences [13], which we overcome by pairing the short 2-bar MusicVAE model with a diffusion model capable of modeling dependencies between $k = 32$ latents, thus modeling 64 bars in total.

MusicVAE embeddings: Each musical phrase is a sequence of one-hot vectors with 16 quantized steps per measure and the vocabulary contains 90 possible tokens (1 note on + 1 note off + 88 pitches). We then parameterize each 2-bar phrase using the pre-trained 2-bar melody MusicVAE [13] and generate a sequence of continuous latent embeddings $z_1, ..., z_k$ to parameterize an entire sequence. The MusicVAE model employs bidirectional recurrent neural networks as an encoder and autoregressive decoding as shown in Figure 4 in the supplement. As we use the pretrained model from the original work, full model details can be found in [13]. After encoding each 2-measure phrase into a latent $z$ embedding, we perform linear feature scaling such that the domain of each embedding is $[-1, 1]$. This ensures consistently scaled inputs starting from the isotropic Gaussian latent $x_N$ for the diffusion model.

Transformer diffusion model: Our network $e_{\theta}(x_t, \sqrt{\alpha_t}); \mathbb{R}^{k \times 42} \times \mathbb{R} \rightarrow \mathbb{R}^{k \times 42}$ is a transformer [18] where $k = 32$ is the length of each sequence of 42-dimensional preprocessed latent embeddings. The unperturbed data distribution used to train the diffusion model is $x_0 = [z_1, ..., z_k]$. The network contains an initial fully-connected layer that projects the embeddings into a 128-dimensional space, followed by $L = 6$ encoder layers each with $H = 8$ self-attention heads and a residual fully-connected layer. All self-attention and residual fully-connected layers which generate the reverse process output. Each fully-connected layer contains 2048 neurons. We use a 128-dimensional sinusoidal positional encoding similar to [18] where $j$ is the position index of a latent input embedding:

$$
\omega = \left[ 10^{-4 \times 3} j, ..., 10^{-4 \times 63} \right], e_j = [\sin(\omega), \cos(\omega)] \quad (8)
$$

This positional encoding $e_1, e_2, ..., e_k$ is added to inputs $x_t$ before being fed through the transformer encoder layers allowing the model to capture the temporal context of the continuous inputs.

Noise schedule and conditioning: As described in both the original diffusion model framework [2] and in [8], we use an additional sinusoidal encoding to condition the diffusion model on a continuous noise level during training and sampling. This noise encoding is identical to the positional encoding described above but with the frequency of each sinusoid scaled by 5000 to account for the updated domain. We use feature-wise linear modulation [20] to generate $\gamma$ (scale) and $\xi$ (shift) parameters given a noise encoding and apply the transformation $\gamma \phi + \xi$ to the output $\phi$ of each layer normalization block in each residual layer, allowing for effective conditioning of the diffusion model. Our model uses a linear noise schedule with $N = 1000$ steps and $\beta_1 = 10^{-8}$ and $\beta_N = 0.01$.

3.2 Unconditional Generation

In the unconditional generation task, the goal is to produce samples that exhibit long-term structure. Because of our multi-stage approach, this works even in the scenario where the KL divergence between the marginal posterior $q_s(z)$ and the Gaussian prior is quite large because the diffusion model accurately captures the structure of the latent space therefore improving the sample quality. Additionally, we extend the underlying VAE to samples longer than what it was trained to model by using the diffusion model to predict sequences of latent embeddings and attempt to generate unconditional samples with coherent patterns across a large number of measures.

3.3 Infilling

One of the benefits of using a sampling process that iteratively refines noise into data samples is that the trajectory of the reverse process can be steered and arbitrarily conditioned without the need for retraining the diffusion model. In creative domains, this post-hoc conditioning is especially useful for artists without the computational resources to modify or re-train deep models for new tasks. We demonstrate the power of diffusion modeling applied to music with conditional infilling of latent embeddings using an unconditionally trained diffusion model.

The infilling procedure extends the sampling procedure described in Algorithm 3 by incorporating information from a partially occluded sample $s$. At each step of sampling, we diffuse the fixed regions of $s$ with the forward process $q(s_t|s) = \mathcal{N}(s_t; \sqrt{\alpha_t} s, (1 - \alpha_t) I)$ and use a mask $m$ to add the diffused fixed regions to the updated sample $x_{t-1}$. The final output $x_0$ will be a version of $s$ with the occluded regions inpainted by the reverse process.

We refer readers to Algorithm 1 for the modified sampling procedure that allows for post-hoc conditional infilling.

Algorithm 1 Infilling

Input: mask $m$, sample $s$, $N$ steps, $\beta_1, ..., \beta_N$

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

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

$e_1, e_2 \sim \mathcal{N}(0, I)$ if $t > 1$, else $e_1 = e_2 = 0$

$y = \sqrt{\alpha_t} s + \sqrt{1 - \alpha_t} e_1$ if $t > 1$, else $s$

$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t}} \epsilon(x_t, \sqrt{\alpha_t}) \right) + \sigma_t e_2$

$x_{t-1} = x_{t-1} \odot (1 - m) + y \odot m$

end for

return $x_0$
4. METHODS

4.1 Data

We use the Lakh MIDI Dataset (LMD) [21] for all experiments. The dataset contains over 170,000 MIDI files with 99% of those files used for training and the remaining used for validation. We extracted 988,893 64-bar monophonic sequences for training and 11,295 for validation from the provided MIDI files. Each sequence was encoded into 32 continuous latent embeddings using MusicVAE. We set the softmax temperature for MusicVAE to 0.001 for decoding generated embeddings in all experiments. A diagram of the MusicVAE architecture used is shown in the supplementary material.

4.2 Autoregressive Baseline

We compare our model to an autoregressive transformer with a mixture density output layer [22] and train on the same dataset as the diffusion model. To ensure a fair comparison, we use the same architecture as our diffusion model with \( L = 6, H = 8, \) and \( K = 2 \) with 2048 neurons for each fully-connected layer. The mixture density layer outputs a mixture of 100 Gaussians to ensure sufficient mode coverage. In total, our baseline model has 38M trainable parameters. While the model is the same as the diffusion model (25.58M trainable parameters) up until the output layer, the autoregressive model has more parameters due to the much larger output layer. We refer to this model as TransformerMDN.

4.3 Training

All models were trained using Adam [23] with default parameters. We trained our diffusion model for 500K steps on a single NVIDIA Tesla V100 GPU for 6.5 hours using a learning rate of \( 10^{-3} \) annealed with a decay rate of 0.98 every 4000 steps and batch size 64. Unlike the diffusion model, which is non-autoregressive, we train TransformerMDN with teacher forcing. We use a batch size of 128, learning rate \( 3 \times 10^{-4} \), and train for 250K steps on a single NVIDIA Tesla V100 GPU for 6.5 hours.

We used the open-source implementation of MusicVAE written in TensorFlow [24] and the publicly available 2-bar melody checkpoints trained on LMD. We trained our diffusion and baseline models \(^2\) with JAX [25] and Flax [26].

4.4 Framewise Self-similarity Metric

To evaluate the statistical similarity between our model’s qualitative output and the original training sequences, we present a metric that captures local self-similarity patterns across generated melodic sequences. Inspired by the statistical similarity evaluation described in [27], we evaluate our models with a modified framewise Overlapping Area (OA) metric.

We use a sliding 4-measure window with 2-measure hop size to capture local pitch and duration statistics across the piece. Within each 4-measure frame, we compute the mean and variance of both pitch, which captures melodic similarity, and duration, which captures rhythmic similarity. These statistics specify a Gaussian PDF for pitch and duration for each frame \((p\_P(k), p\_D(k))\). We compute the Overlapping Area (OA) [27] of adjacent frames \((k, k + 1)\) where each frame’s statistics are modeled as \( N(\mu_1, \sigma_1^2) \) and \( N(\mu_2, \sigma_2^2) \), respectively:

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

for both pitch \((OA_P)\) and duration \((OA_D)\), where \( \text{erf} \) is the Gauss error function and \( c \) is the point of intersection between Gaussian PDFs with \( \mu_1 < \mu_2 \). For a set of MIDI samples, we infer the Consistency and Variance from the mean \((\mu_{OA})\) and variance \((\sigma_{OA}^2)\) respectively of OA aggregated over all adjacent frames. We then use these aggregate values to compute the normalized relative similarity of pitch and duration consistency and variance to the training set (GT):

\[
\text{Consistency} = \max(0, 1 - \frac{|\mu_{OA} - \mu_{GT}|}{\mu_{GT}}) \\
\text{Variance} = \max(0, 1 - \frac{|\sigma_{OA}^2 - \sigma_{GT}^2|}{\sigma_{GT}^2})
\]

We clip Consistency and Variance such that samples with \( \mu_{OA} \) or \( \sigma_{OA}^2 \) with greater than 100% percent error from the ground truth are considered to have zero relative similarity.

4.5 Latent Space Evaluation

We evaluate the similarity of latent embeddings generated by each of our models using the Fréchet distance (FD) [28] and Maximum Mean Discrepancy (MMD) [29] with a polynomial kernel, which are popular evaluation metrics in the generative modeling literature. These metrics measure the distance between the models’ continuous output distributions and the original data distribution in latent space. It is important to note that this metric does not measure long-term temporal consistency or quality of produced sequences and only measures the quality of the intermediate continuous representation before the final sequence is generated using the MusicVAE decoder.

5. RESULTS

5.1 Unconditional Generation

To evaluate unconditional sample quality, we compare batches of 1000 samples (32 latents each) generated by our proposed diffusion model with random draws from the training and test sets. We compare against a set of baseline generators including TransformerMDN, the independent \( N(0, I) \) MusicVAE prior, and spherical interpolation [30] between two MusicVAE embeddings at the start and end of an example from the test set. As seen in Table 1, the diffusion model quantitatively produces samples most similar to the training data according to the relative framewise overlapping area metrics for...
Figure 2. Example piano rolls of infilling experiments. For each example (A, B) the first and last 256 melody tokens are held constant, and the interior 512 tokens are filled in by the model (dashed red box). Even qualitatively, it is visually apparent that the diffusion model (second row) produces notes with a consistency and variance similar to the original data (first row), while the latent interpolation (third row) is too repetitive, and sampling independently from the prior (last row) produces outputs with too much variety and lack of local coherence.

Table 1. Framewise self-similarity of consistency (C) and variance (Var), as defined in Equation 10, for note pitch and duration. For both unconditional sampling and infilling tasks, the diffusion model produces samples most similar to the real data. For diffusion samples, we use $N = 1000$ sampling steps with $\beta_1 = 10^{-6}$ and $\beta_N = 0.01$. For the TransformerMDN baseline we sample with a temperature of 1.0, meaning we sampled directly from the logits of mixture density layer. Absolute values of overlap area can be found in Table 2 in our supplementary material.

```

<table>
<thead>
<tr>
<th>Setting</th>
<th>Unconditional</th>
<th>Infilling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>Pitch</td>
<td>Duration</td>
</tr>
<tr>
<td>Train Data</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Test Data</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>Diffusion</td>
<td>0.99</td>
<td>0.90</td>
</tr>
<tr>
<td>Autoregression</td>
<td>0.93</td>
<td>0.68</td>
</tr>
<tr>
<td>Interpolation</td>
<td>0.85</td>
<td>0.23</td>
</tr>
<tr>
<td>$N(0, I)$ Prior</td>
<td>0.84</td>
<td>0.19</td>
</tr>
</tbody>
</table>
```

Figure 3 helps us to further understand the iterative refinement process by showing the improvement in sample quality as a function of iterative refinement time for both latent space and framewise self-similarity metrics. Interestingly, latent metrics improve steadily, while consistency similarity starts fairly high, and variance similarity only emerges at the end of refinement. We refer the reader to the supplementary material for extended visual and audio evaluations.
The quantitative evaluations in Table 1 back up these observations. Similar to the unconditional generation task, the diffusion model outperforms the baselines in both consistency and variance similarity. We do not include the autoregressive baseline because it is unable to condition on the final 8 embeddings.

5.2 Infilling

To probe the diffusion model’s ability to perform post-hoc conditional generation, we remove the middle 32 measures (16 embeddings) and generate new embeddings following Algorithm 1 by conditioning on the first and last 8 embeddings. As in the unconditional setting, we compare to interpolation and independent samples from the prior.

Figure 2 visualizes the task by plotting the resulting note sequences for two different examples. Even qualitatively, it is visually apparent that the diffusion model produces notes with a consistency and variance similar to the original data. Latent interpolation is very consistent but unrealistically repetitive, and sampling independently from the prior produces a sequence with extremely large variance that is inconsistent in both pitch and duration.

The quantitative evaluations in Table 1 back up these observations. Similar to the unconditional generation task, the diffusion model outperforms the baselines in both consistency and variance similarity. We do not include the autoregressive baseline because it is unable to condition on the final 8 embeddings.

6. RELATED WORK

Multi-stage learning: Several models have achieved long-term structure by first modeling some intermediate representation and then using that to guide the final generative process. Wave2Midi2Wave [31] uses a transformer to generate MIDI-like symbolic data and then a WaveNet [32] to synthesize that symbolic data into audio. Jukebox [11] and DALL-E [12] use similar approaches for text-conditioned generative music audio and image models. There has also been work investigating the extension of a single-measure VAE to multiple measures by using an autoregressive LSTM with a mixture density output layer [33], similar to TransformerMDN.

Iterative refinement: [34] use an orderless NADE with blocked Gibbs sampling to iteratively rewrite musical harmonies based on surrounding context. [35] use a gradient-based sampler combined with a restricted Boltzmann machine to generate polyphonic piano music. Similarly, [36] investigated the use of a score-based generative model [5] to generate Bach chorales with annealed Langevin dynamics. A key distinction between our method and prior work is the use of a VAE to parameterize the discrete space of musical notes for improved generation with a DDPM while previous methods have performed iterative refinement in the discrete space directly.

Conditional sampling from unconditional models: We build on top of the breadth of material that investigate steering generation in the latent space of an unconditionally trained generative model [37–40]. Most similar to our work is [4,37,41] which train an additional model on top of a pre-trained variational autoencoder to steer generation in the latent space of that autoencoder. Our approach builds on top of this by not only improving sampling and generation of the underlying autoencoder but also extending generation to sequences much longer than those used to train the VAE. We also use a diffusion model to refine generation and provide conditional infilling while the works mentioned primarily used conditional GANs and VAEs to extend the underlying autoencoder for a single latent embedding as opposed to a sequence of latent embeddings.

7. CONCLUSION

We have proposed and demonstrated a multi-stage generative model comprised of a low-level variational autoencoder with continuous latents modeled by a higher-level diffusion model. This approach enables using diffusion models on discrete data, and as priors for modeling long-term structure in multi-stage systems. We demonstrate that this model is useful for symbolic music generation, both in unconditional generation and conditional infilling. Future work will include extending this approach to other discrete data such as text, and exploring a greater array of approaches for post-hoc conditioning in creative applications.

Figure 3. Sample quality improvement during iterative refinement. Latent space and framewise metrics evaluated at different stages of unconditional sampling. The metrics improve as the iterative refinement process progresses. We plot the means and standard deviations for 1000 samples.

3https://goo.gl/magenta/symbolic-music-diffusion-examples
8. ACKNOWLEDGEMENTS

We thank Anna Huang, Carrie Cai, Sander Dieleman, Doug Eck and the rest of the Magenta team for helpful discussions, feedback, and encouragement.

9. REFERENCES


