SYMBOLIC MUSIC GENERATION WITH DIFFUSION MODELS

Gautam Mittal[1⋆] **Jesse Engel**[2] **Curtis Hawthorne**[2] **Ian Simon**[2]
[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 $x_1, ..., x_N$ 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:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \qquad (1)$$

$$q(x_{1:N}|x_0) = \prod_{t=1}^{N} q(x_t|x_{t-1}) \qquad (2)$$

where $\beta_1, \beta_2, ..., \beta_N$ 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 θ that iteratively refines latent point $x_N \sim \mathcal{N}(0, I)$ into data point x_0. The learned transition probabilities are defined as,

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

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

where the objective is to gradually denoise samples at each reverse diffusion step t. In practice, σ_θ is set to an untrained time-dependent constant based on the noise schedule, and [2] found $\sigma_\theta(x_t, t) = \sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ to have reasonable practical results, where $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

[⋆] Work completed during an internship at Google Brain.

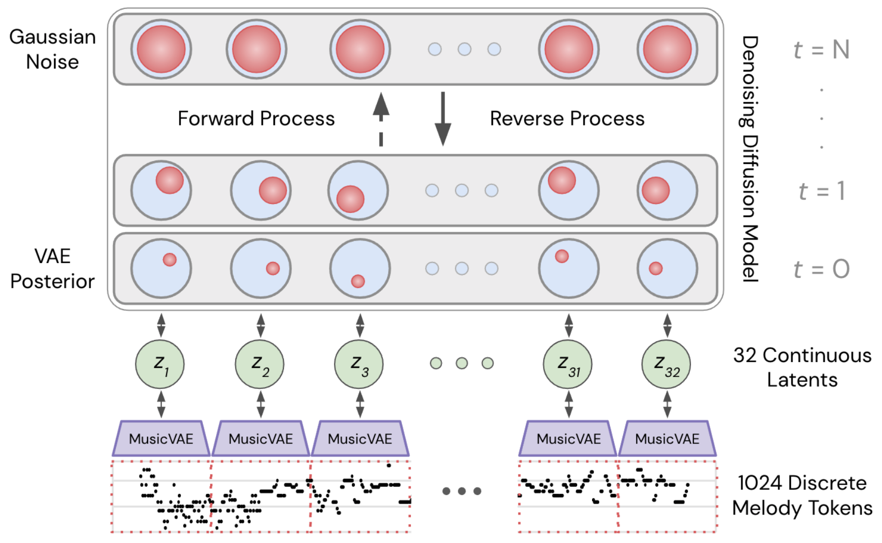

Figure 1. A diagram of our proposed framework. We use the pre-trained 2-bar melody MusicVAE [13] to embed discrete musical phrases (64 bars, 1024 tokens) into a sequence of continuous latent codes (32 latents, 512 dimensions each). These embeddings are used to train a diffusion model that iteratively adds noise such that after N diffusion steps the input embeddings are distributed $\mathcal{N}(0, I)$. To sample from this model, we initialize Gaussian noise and use the reverse process to iteratively refine the noise samples into a sequence of embeddings from the data distribution. These generated embeddings are fed through the MusicVAE decoder to produce the final MIDI sequence.

The training objective is to maximize the log likelihood of $p_\theta(x_0) = \int p_\theta(x_0, ..., x_N) dx_{1:N}$, but the intractability of this marginalization leads to the following evidence lower bound (ELBO):

$$\begin{aligned}\mathbb{E}\left[\log p_\theta(x_0)\right] &\geq \mathbb{E}_q\left[\log \frac{p_\theta(x_{0:N})}{q(x_{1:N}|x_0)}\right] \\ &= \mathbb{E}_q\left[\log p(x_N) + \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]\end{aligned} \tag{5}$$

Additionally, [2] observed that the forward process can be computed for any step t such that $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$, 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:

$$L(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right] \tag{6}$$

where t is sampled uniformly between 1 and N, $\epsilon \sim \mathcal{N}(0, I)$, and ϵ_θ 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 $\sqrt{\bar{\alpha}} \sim \mathbb{U}(\sqrt{\bar{\alpha}_{t-1}}, \sqrt{\bar{\alpha}_t})$ where $t \sim \mathbb{U}(\{1, ..., N\})$ and $\bar{\alpha}_0 = 1$.

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:

$$\mathbb{E}\left[\log p_\theta(y|z)\right] - \mathrm{KL}(q_\gamma(z|y)||p(z)) \leq \log p(y) \tag{7}$$

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 supplement 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 γ 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 $\epsilon_\theta(x_t, \sqrt{\bar{\alpha}}) : \mathbb{R}^{k\times 42} \times \mathbb{R} \to \mathbb{R}^{k\times 42}$ 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 fully-connected layers use layer normalization [19]. The output of the encoder is fed to $K = 2$ noise-conditioned 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:

$$\omega = \left[10^{\frac{-4\times 0}{63}} j, ..., 10^{\frac{-4\times 63}{63}} j\right] \quad e_j = [\sin(\omega), \cos(\omega)] \quad (8)$$

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 γ (scale) and ξ (shift) parameters given a noise encoding and apply the transformation $\gamma\phi + \xi$ to the output ϕ 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 $\beta_1 = 10^{-6}$ 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 $q_\gamma(z)$ 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 $q(s_t|s) = \mathcal{N}(s_t; \sqrt{\bar{\alpha}_t}s, (1-\bar{\alpha}_t)I)$ 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
    $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, I)$ if $t > 1$, else $\epsilon_1 = \epsilon_2 = 0$
    $y = \sqrt{\bar{\alpha}_t}s + \sqrt{1-\bar{\alpha}_t}\epsilon_1$ if $t > 1$, else $s$
    $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, \sqrt{\bar{\alpha}_t})\right) + \sigma_t\epsilon_2$
    $x_{t-1} = x_{t-1} \odot (1-m) + y \odot m$
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 3×10^{-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 $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, respectively:

$$\mathrm{OA}(k, k+1) = 1 - \mathrm{erf}\left(\frac{c - \mu_1}{\sqrt{2}\sigma_1^2}\right) + \mathrm{erf}\left(\frac{c - \mu_2}{\sqrt{2}\sigma_2^2}\right) \quad (9)$$

for both pitch (OA_P) and duration (OA_D), where 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 (μ_{OA}) and variance (σ^2_{OA}) 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):

$$\begin{aligned} Consistency &= \max(0, 1 - \frac{|\mu_{\mathrm{OA}} - \mu_{GT}|}{\mu_{GT}}) \\ Variance &= \max(0, 1 - \frac{|\sigma^2_{\mathrm{OA}} - \sigma^2_{GT}|}{\sigma^2_{GT}}) \end{aligned} \quad (10)$$

We clip *Consistency* and *Variance* such that samples with μ_{OA} or σ^2_{OA} 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 $\mathcal{N}(0, I)$ 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

[2] Our implementation is available at `https://github.com/magenta/symbolic-music-diffusion`

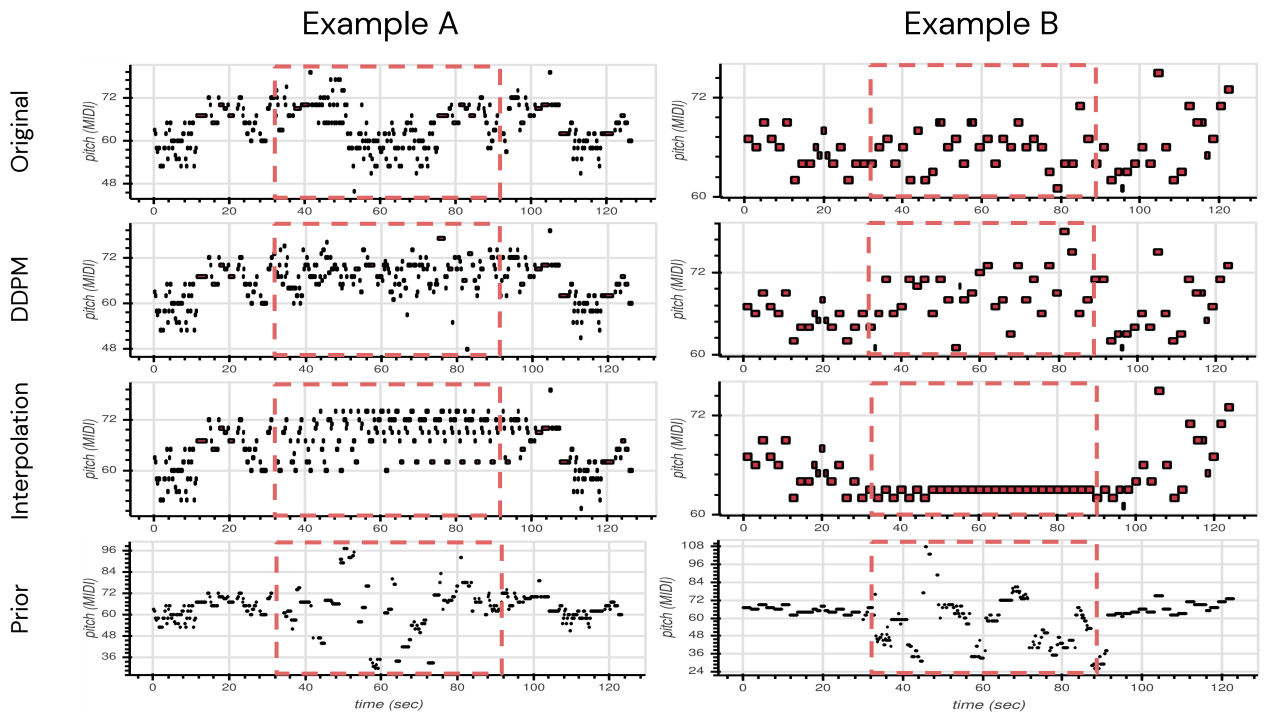

Figure 2. Example piano rolls of infilling experiments. For each example (A, B) the first and last 256 melody tokens are held constant, and the interior 512 tokens are filled in by the model (dashed red box). Even qualitatively, it is visually apparent that the diffusion model (second row) produces notes 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.

Setting	Unconditional				Infilling			
Quantity	Pitch		Duration		Pitch		Duration	
Metric	C	Var	C	Var	C	Var	C	Var
Train Data	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Test Data	1.00	0.96	1.00	0.91	1.00	0.96	1.00	0.91
Diffusion	**0.99**	**0.90**	**0.96**	**0.92**	**0.97**	**0.87**	**0.97**	**0.80**
Autoregression	0.93	0.68	0.93	0.76	-	-	-	-
Interpolation	0.85	0.23	0.91	0.34	0.94	0.78	0.96	**0.80**
$\mathcal{N}(0, I)$ Prior	0.84	0.19	0.90	0.67	0.89	0.19	0.94	0.54

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.

note pitch and duration. The diffusion model outperforms TransformerMDN, which is challenged by modelling the relatively high-dimensional continuous latents autoregressively, even with a mixture of Gaussians output. The autoregressive models are also trained with teacher forcing that results in exposure bias, leading to divergence from the data distribution during sampling. This is reflected in the lower pitch and duration consistency and higher variance in the absolute overlapping area numbers seen in Supplemental Table 2. Additionally, the diffusion model is able to capture the joint dependencies of the sequences better because it learns to model all latents simultaneously as opposed to autoregressively. Note that the Gaussian prior also suffers from low consistency and high variance, due to lack of temporal dependencies, while the interpolated samples conversely suffer from low variance and too much consistency due to high repetition.

Table 3 in our supplement presents latent space evaluations of our generated samples. The TransformerMDN outperformed all other models, likely due to the Gaussian mixture prior on its output layer whereas the diffusion model must learn the output distribution from scratch. Furthermore, the latent space metrics are limited by assumptions about the latent manifold distribution and are unable to fully capture the detail of the entire space, further highlighting the necessity of our quantitative framewise self-similarity metrics and qualitative evaluations.

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

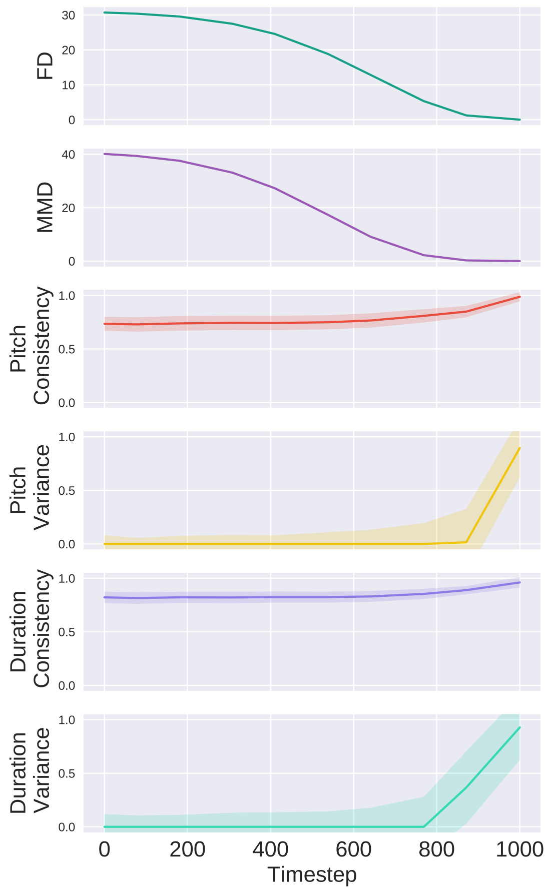

Figure 3. Sample quality improvement during iterative refinement. Latent space and framewise metrics evaluated at different stages of unconditional sampling. The metrics improve as the iterative refinement process progresses. We plot the means and standard deviations for 1000 samples.

samples of the generated sequences from each model [3] .

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.

[3] `https://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

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265. [Online]. Available: http://proceedings.mlr.press/v37/sohl-dickstein15.html

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[3] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 681–688.

[4] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sy8XvGb0-

[5] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 895–11 907.

[6] ——, "Improved techniques for training score-based generative models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[7] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf

[8] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=NsMLjcFaO8O

[9] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=a-xFK8Ymz5J

[10] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf

[11] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[12] A. Ramesh, M. Pavlov, G. Goh, S. Gray, M. Chen, R. Child, V. Misra, P. Mishkin, G. Krueger, S. Agarwal, and I. Sutskever, "Dall·e: Creating images from text," *OpenAI blog*, 2021. [Online]. Available: https://openai.com/blog/dall-e

[13] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4364–4373. [Online]. Available: http://proceedings.mlr.press/v80/roberts18a.html

[14] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

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

[16] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," 2016.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[20] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[21] C. Raffel, “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching,” Ph.D. dissertation, Columbia University, 2016.

[22] C. M. Bishop, “Mixture density networks,” 1994.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[25] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018. [Online]. Available: http://github.com/google/jax

[26] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, “Flax: A neural network library and ecosystem for JAX,” 2020. [Online]. Available: http://github.com/google/flax

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

[28] 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.

[29] 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

[30] T. White, “Sampling generative networks,” *arXiv preprint arXiv:1609.04468*, 2016.

[31] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7

[32] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.

[33] H. Jhamtani and T. Berg-Kirkpatrick, “Modeling self-repetition in music generation using generative adversarial networks,” 2019.

[34] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” in *Proceedings of ISMIR 2017*, 2017. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/187_Paper.pdf

[35] S. Lattner, M. Grachten, and G. Widmer, “Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints,” *arXiv preprint arXiv:1612.04742*, 2016.

[36] E. Zhang and R. Sirohi, “Generative modeling of bach chorales by gradient estimation,” 2020. [Online]. Available: https://www.ekzhang.com/assets/pdf/Generative_Music_Modeling.pdf

[37] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, “Plug & play generative networks: Conditional iterative generation of images in latent space,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4467–4477.

[38] N. Jaques, J. Engel, D. Ha, F. Bertsch, R. Picard, and D. Eck, “Learning via social awareness: improving sketch representations with facial feedback,” in *International Conference on Learning Representations*, 2018, workshop Track. [Online]. Available: https://arxiv.org/abs/1802.04877

[39] A. Jahanian, L. Chai, and P. Isola, “On the" steerability" of generative adversarial networks,” *arXiv preprint arXiv:1907.07171*, 2019.

[40] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, “Plug and play language models: A simple approach to controlled text generation,” *arXiv preprint arXiv:1912.02164*, 2019.

[41] M. Dinculescu, J. Engel, and A. Roberts, Eds., *MidiMe: Personalizing a MusicVAE model with user data*, 2019.

[42] M. D. Hoffman and M. J. Johnson, “Elbo surgery: yet another way to carve up the variational evidence lower bound.”

[43] A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy, “Fixing a broken elbo,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 159–168.

A. DDPM TRAINING AND SAMPLING

To train and sample from our diffusion model, we use the algorithms as described in [8].

Algorithm 2 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 3 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** $x_0$
```

B. MUSICVAE ARCHITECTURE

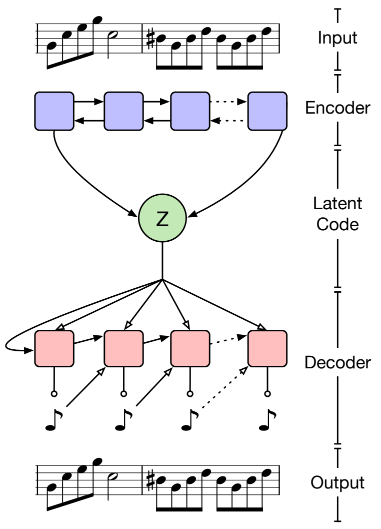

Figure 4. 2-bar melody MusicVAE architecture. The encoder is a bi-direction LSTM and the decoder is an autoregressive LSTM.

C. TRIMMING LATENTS

During training VAEs typically learn to only utilize a fraction of their latent dimensions. As shown in Figure 5, by examining the standard deviation per dimension of the posterior $q(z|y)$ averaged across the entire training set, we are able to identify underutilized dimensions where the average embedding standard deviation is close to the prior of 1. The VAE loss encourages the marginal posterior to match to the prior [42,43], but to encode information, dimensions must have smaller variance per an example.

In all experiments, we remove all dimensions except for the 42 dimensions with standard deviations below 1.0, before training the diffusion model on the input data. We find this latent trimming to be essential for training as it helps to avoid modeling unnecessary high-dimensional noise and is very similar to the distance penalty described in [4]. We also tried reducing the dimensionality of embeddings with principal component analysis (PCA) but found that the lower dimensional representation captured too many of the noisy dimensions and not those with high utilization.

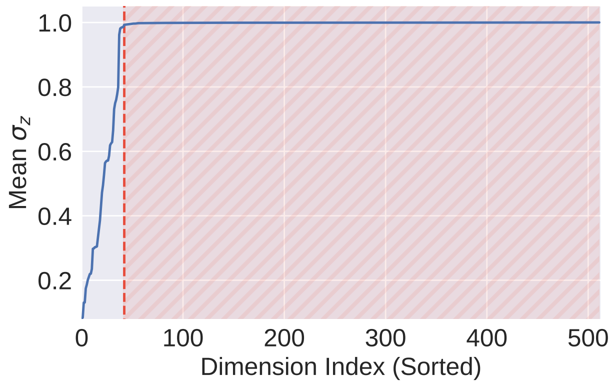

Figure 5. The standard deviation per dimension of the MusicVAE posterior $q(z|y)$ averaged across the entire training set. The region highlighted in red contains the latent dimensions that are unused.

D. TABLES

In Tables 2 and 3, we present the unnormalized framewise self-similarity results as well as the latent space evaluation of each model.

Setting	Unconditional				Infilling			
Quantity	Pitch		Duration		Pitch		Duration	
OA	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
Train Data	0.82	0.018	0.88	0.012	0.82	0.018	0.88	0.012
Test Data	0.82	0.018	0.88	0.011	0.82	0.018	0.88	0.011
Diffusion	**0.81**	**0.017**	**0.85**	**0.013**	**0.80**	**0.021**	**0.86**	**0.015**
Autoregression	0.76	0.024	0.82	0.015	-	-	-	-
Interpolation	0.94	0.004	0.96	0.004	0.87	0.014	0.91	**0.009**
$\mathcal{N}(0, I)$ Prior	0.69	0.033	0.79	0.016	0.73	0.033	0.82	0.018

Table 2. Unnormalized framewise self-similarity (overlapping area) evaluation of unconditional and conditional samples. Evaluations of same samples as in Table 1. Note the interpolations have unrealistically high mean overlap and low variance, while the Gaussian prior and TransformerMDN samples suffer from unrealistically lower mean overlap and higher variance.

Setting	Unconditional		Infilling	
Metric	FD$\times 10^{-2}$	MMD$\times 10^{-2}$	FD$\times 10^{-2}$	MMD$\times 10^{-2}$
Train Data	0.00	0.00	0.00	0.00
Test Data	1.24	0.12	1.24	0.12
Diffusion	1.66	0.18	1.53	0.16
Autoregression	**1.26**	**0.12**	-	-
Interpolation	3.22	0.43	1.97	0.23
$\mathcal{N}(0, I)$ Prior	2.44	0.29	**1.17**	**0.12**

Table 3. Latent space evaluation of infilling and unconditional and conditional samples. As described in Section 4.5, the TransformerMDN performs better in latent space similarity, even while producing less realistic samples (as seen in Tables 1 and 2).

E. ADDITIONAL SAMPLES

In Figure 6 we provide piano rolls of sequences drawn from the test set and in Figures 7, 8, 9, and 10 we present additional samples unconditionally generated by our diffusion model, TransformerMDN, spherical interpolation, and through independent sampling from the MusicVAE prior, respectively. Additional piano roll visualizations from infilling experiments are provided in Figure 11.

For extended visual and audio samples of the generated sequences from each model, we refer the reader to the online supplement available at `https://goo.gl/magenta/symbolic-music-diffusion-examples`.

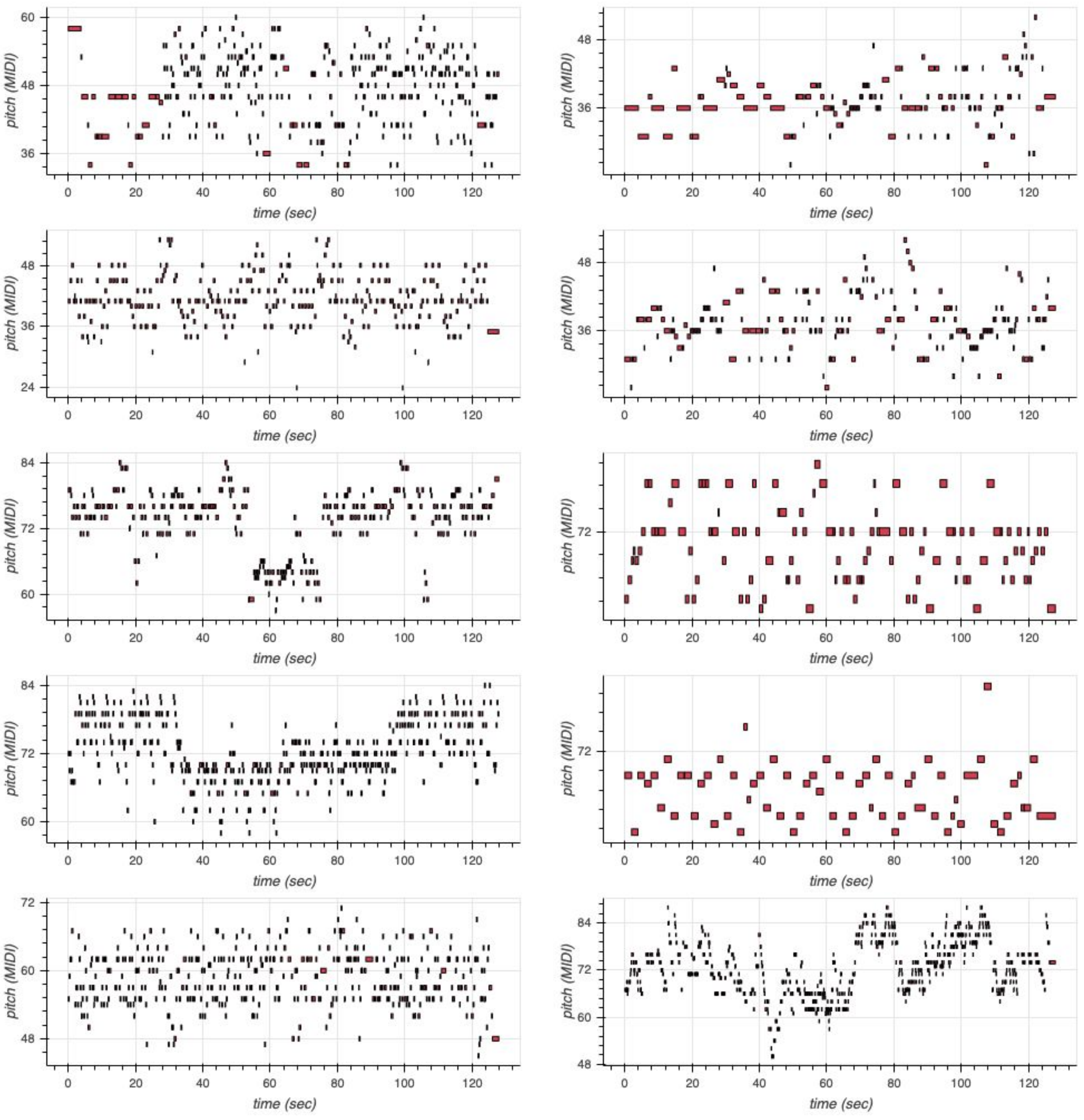

Figure 6. Additional piano rolls from the test set.

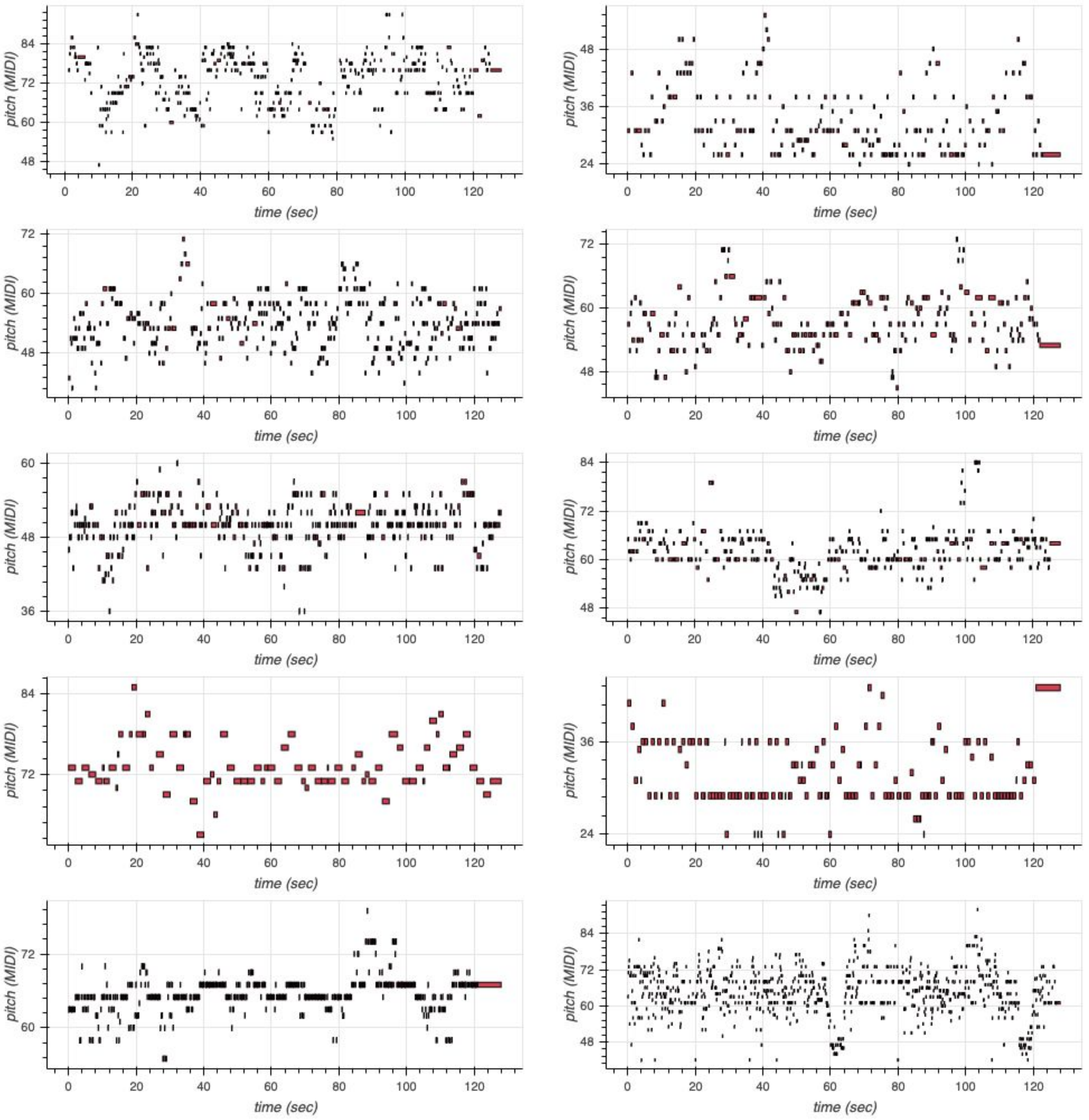

Figure 7. Additional piano rolls generated unconditionally by our diffusion model.

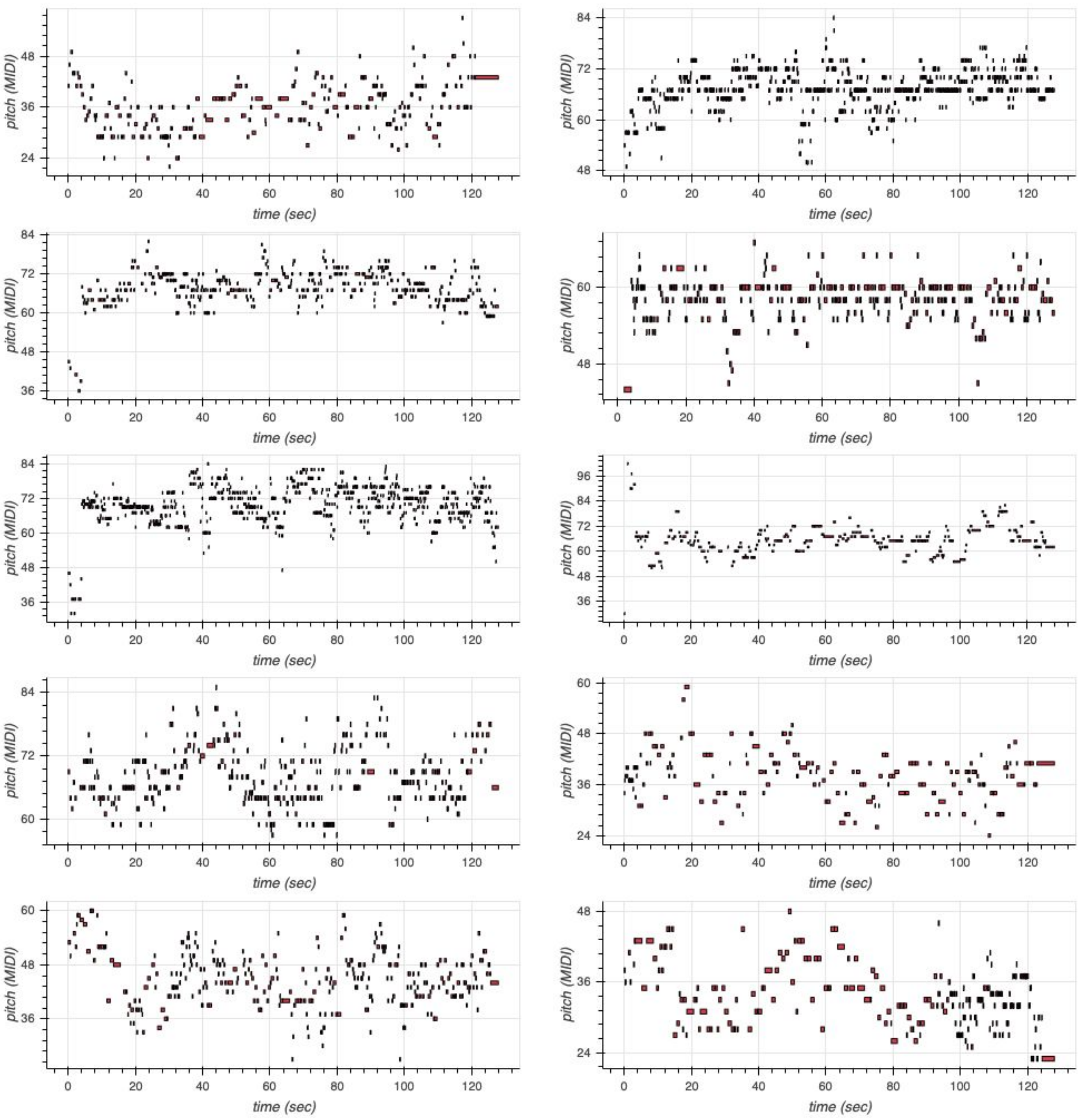

Figure 8. Additional piano rolls generated unconditionally by TransformerMDN.

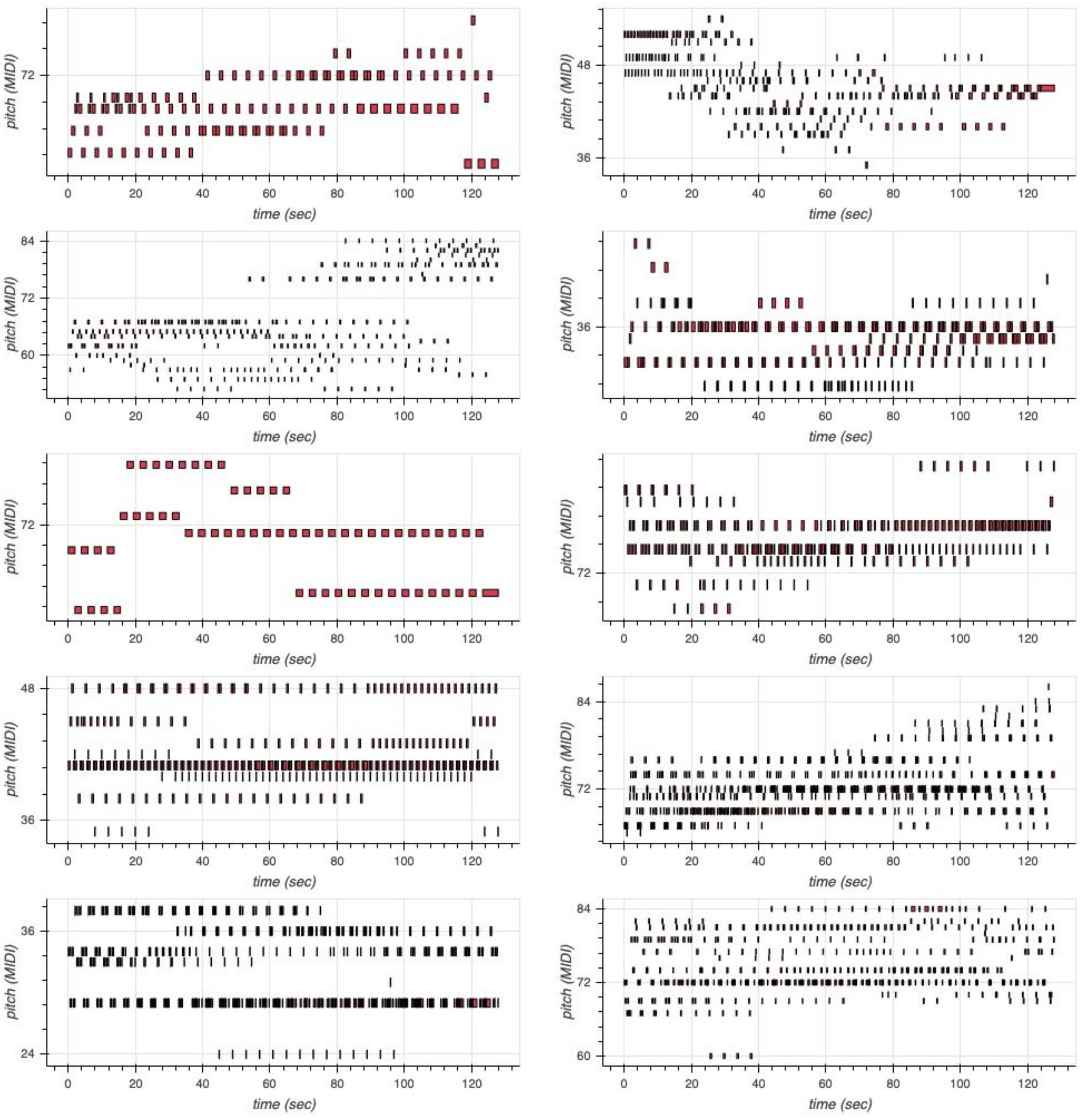

Figure 9. Additional piano rolls generated by performing spherical interpolation [30] between the first and last latent embeddings of sequences drawn from the test set.

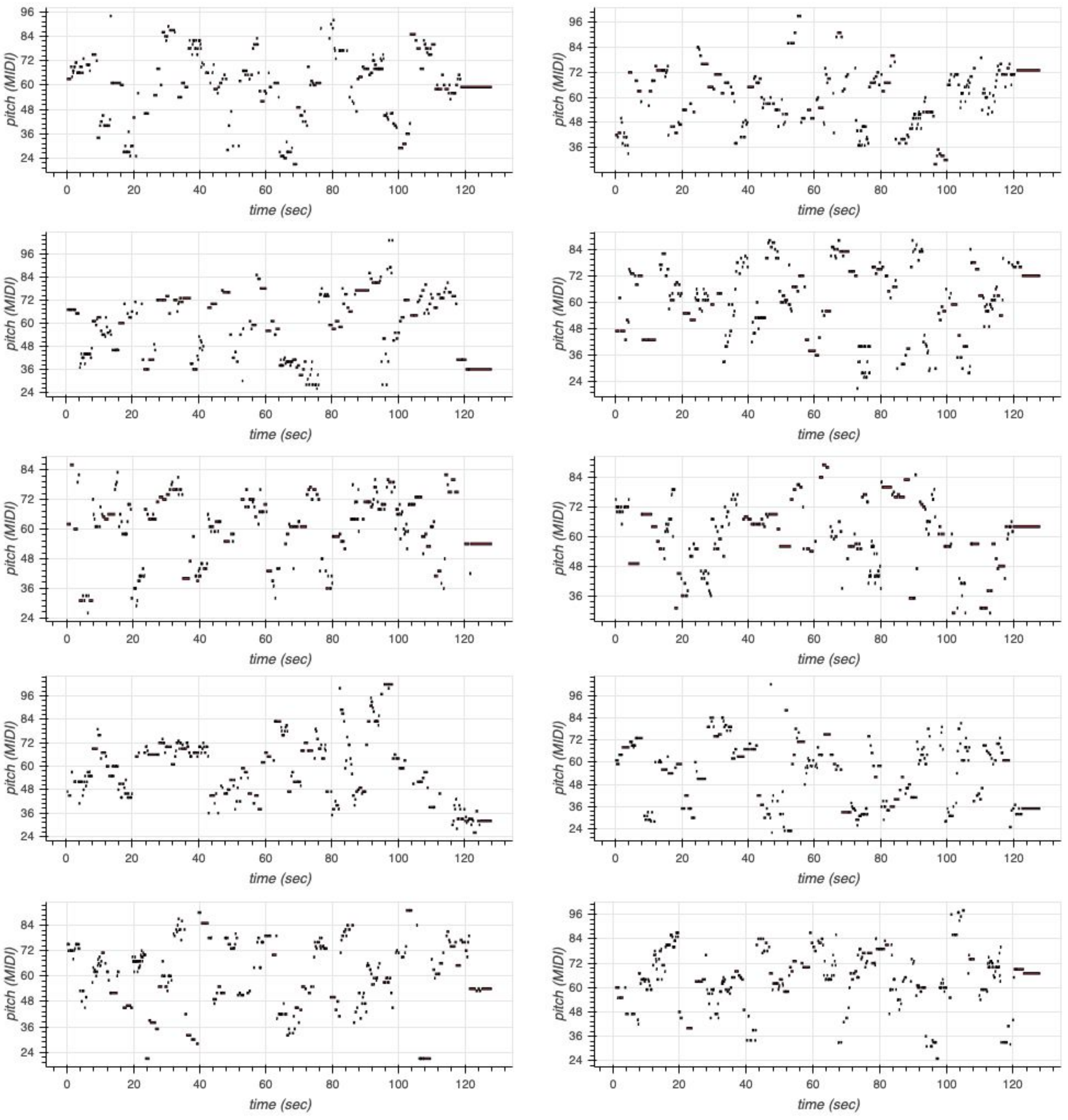

Figure 10. Additional piano rolls generated by sampling each latent embedding independently from the $\mathcal{N}(0, I)$ MusicVAE prior.

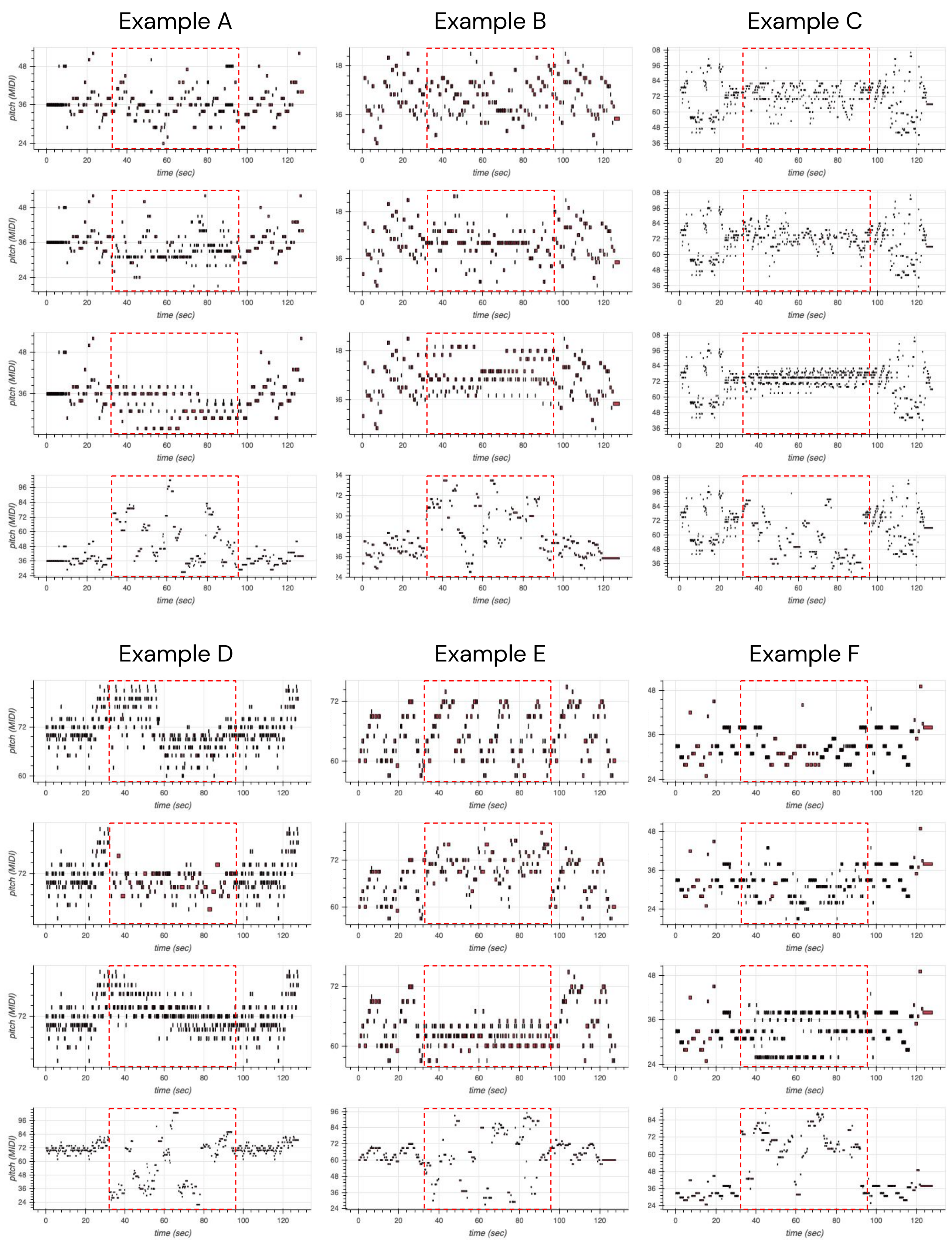

Figure 11. Additional piano rolls of infilling experiments. The first and last 256 melody tokens are held constant and the interior 512 tokens are filled in by the model (dashed red box). Original sample (first row), diffusion model (second row), interpolation (third row), sampling independently from the MusicVAE prior (fourth row).