

A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music

Adam Roberts¹ Jesse Engel¹ Colin Raffel¹ Curtis Hawthorne¹ Douglas Eck¹

Abstract

The Variational Autoencoder (VAE) has proven to be an effective model for producing semantically meaningful latent representations for natural data. However, it has thus far seen limited application to sequential data, and, as we demonstrate, existing recurrent VAE models have difficulty modeling sequences with long-term structure. To address this issue, we propose the use of a *hierarchical* decoder, which first outputs embeddings for subsequences of the input and then uses these embeddings to generate each subsequence independently. This structure encourages the model to utilize its latent code, thereby avoiding the “posterior collapse” problem which remains an issue for recurrent VAEs. We apply this architecture to modeling sequences of musical notes and find that it exhibits dramatically better sampling, interpolation, and reconstruction performance than a “flat” baseline model. An implementation of our “MusicVAE” is available online.¹

1. Introduction

Generative modeling describes the framework of estimating the underlying probability distribution $p(x)$ used to generate data x . This can facilitate a wide range of applications, from sampling novel datapoints to unsupervised representation learning to estimating the probability of an existing datapoint under the learned distribution. Much recent progress in generative modeling has been expedited by the use of deep neural networks, producing “deep generative models,” which leverage the expressive power of deep networks to model complex and high-dimensional distributions. Practical achievements include generating realistic images with

¹Google Brain, Mountain View, CA, USA. Correspondence to: Adam Roberts <adarob@google.com>.

¹Colab: <http://g.co/magenta/musicvae-colab>
Implementation: https://github.com/tensorflow/magenta/tree/master/magenta/models/music_vae/

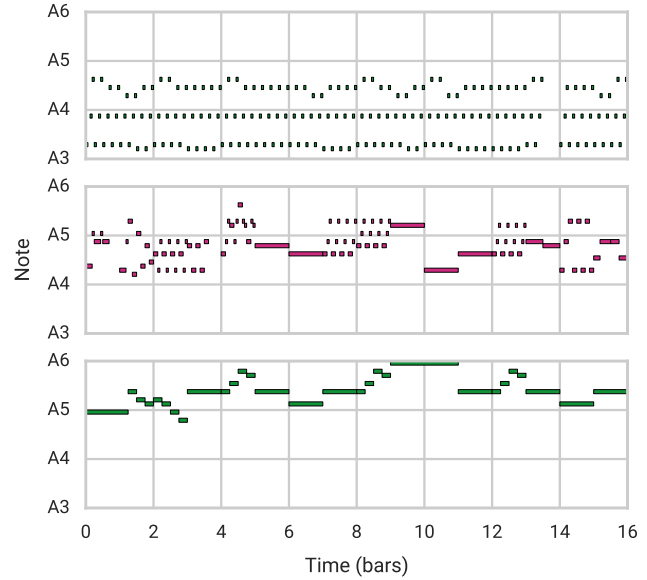


Figure 1. Demonstration of latent-space averaging using the MusicVAE. The latent codes for the top and bottom sequences are averaged and decoded by our model to produce the middle sequence. The latent-space mean involves a similar repeating pattern to the top sequence, but in a higher register and with intermittent pauses like the bottom sequence. See supplemental Figures 9, 10 for baseline comparisons.

millions of pixels (Karras et al., 2017), generating synthetic audio with hundreds of thousands of timesteps (van den Oord et al., 2016a), and achieving state-of-the-art performance on semi-supervised learning tasks (Wei et al., 2018).

A wide variety of model structures have been used in deep generative modeling, including implicit models such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and explicit deep autoregressive models such as PixelCNN (van den Oord et al., 2016b) and WaveNet (van den Oord et al., 2016a). In this work, we are primarily focused on deep latent variable models such as the Variational Autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014). The advantage of these models is that they explicitly model both $p(z|x)$ and $p(z)$, where z is a latent vector which can therefore either be inferred from existing data or sampled from a distribution over the latent space.

Ideally, the latent vector captures the pertinent characteristics of a given datapoint and disentangles factors of variation in a dataset. These autoencoders also model the likelihood $p(x|z)$ which provides an efficient way of mapping the latent vector back to the data space.

Our interest in deep latent variable models primarily comes from their increasing use in creative applications of machine learning (Carter & Nielsen, 2017; Ha & Eck, 2018; Engel et al., 2017a). This arises from surprising and convenient characteristics of the latent spaces typically learned by these models. For example, averaging the latent codes for all datapoints which possess a given attribute produces a so-called attribute vector. Encoding a datapoint (say, a photograph of a person with brown hair) to obtain its latent space representation, subtracting an attribute vector corresponding to that datapoint (the “brown hair” vector), adding another attribute vector (“blond hair”), and decoding the resulting latent vector can produce a realistic realization of the initial point with the attribute swapped (the same person with blond hair) (Larsen et al., 2016; Mikolov et al., 2013). As another example, interpolating between latent vectors and decoding each point on the trajectory can produce realistic datapoints which vary in a smooth and semantically meaningful way.

Most work on deep latent variable models has focused on modeling continuously-valued data with a fixed dimensionality, e.g., images. Less common is to model sequential data, particularly sequences of discrete tokens such as music data. This is partially caused by the fact that modeling sequential data suggests the use of an autoregressive decoder, which is often sufficiently powerful that the autoencoder simply ignores the latent code (Bowman et al., 2016). While they have shown some success on short sequences (e.g., sentences), deep latent variable models have yet to be successfully applied to very long sequences.

To address this gap, we introduce a novel sequential autoencoder with a hierarchical recurrent decoder, which helps overcome the aforementioned issue of modeling long-term structure with recurrent VAEs. Our model encodes an entire sequence to a single latent vector, which enables many of the creative applications enjoyed by VAEs of images. We show experimentally that our model is capable of effectively autoencoding substantially longer sequences than a baseline model with a “flat” decoder RNN.

In this paper, we focus on the application of modeling sequences of musical notes. Western popular music exhibits strong long-term structure, such as the repetition and variation between measures and sections of a piece of music. This structure is also hierarchical – songs are divided into sections, which are broken up into measures, and then into beats, and so on. Further, music is fundamentally a multi-stream signal, in the sense that it often involves multiple players generating music with strong inter-player dependen-

cies. These unique properties, in addition to the potential creative applications, make music an ideal testbed for our sequential autoencoder.

After covering a background of related work our approach builds on, we describe our model and its novel architectural enhancements. We then provide an overview of related work towards applying latent variable models to sequences. Finally, in section 5, we apply our model to musical data and carry out various evaluations to qualitatively and quantitatively demonstrate its abilities.

2. Background

Fundamentally, our model is an autoencoder, i.e., its goal is to accurately reconstruct its inputs. However, we would additionally like to be able to draw novel samples from our model and perform latent-space interpolations and attribute vector arithmetic. For these properties, we adopt the framework of the Variational Autoencoder (VAE). Successfully using VAEs for sequences benefits from some additional extensions to the VAE objective. In the following subsections, we cover the prior work which forms the backbone for our approach.

2.1. Variational Autoencoders

A common constraint applied to autoencoders is that they compress the relevant information about the input into a lower-dimensional latent code. Ideally, this forces the model to produce a compressed representation which captures important factors of variation in the dataset. In pursuit of this goal, the Variational Autoencoder (Kingma & Welling, 2014; Rezende et al., 2014) (VAE) introduces the constraint that the latent code z is a random variable distributed according to a prior $p(z)$. The data generation model is then $z \sim p(z), x \sim p(x|z)$. The VAE consists of an encoder $q_\lambda(z|x)$, which approximates the posterior $p(z|x)$, and a decoder $p_\theta(x|z)$, which parameterizes the likelihood $p(x|z)$. In practice, the approximate posterior distribution and likelihood distribution (“encoder” and “decoder”) are parameterized by neural networks with parameters λ and θ respectively. Following the framework of Variational Inference, we do posterior inference by minimizing the KL divergence between our approximate posterior, the encoder, and the true posterior $p(z|x)$ by maximizing the evidence lower bound (ELBO)

$$\mathbb{E}[\log p_\theta(x|z)] - \text{KL}(q_\lambda(z|x)||p(z)) \leq \log p(x) \quad (1)$$

where the expectation is taken with respect to $z \sim q_\lambda(z|x)$ and $\text{KL}(\cdot||\cdot)$ is the KL-divergence. Naively computing the gradient through the ELBO is infeasible due to the sampling operation used to obtain z . In the common case where $p(z)$ is a diagonal-covariance Gaussian, this can be circumvented

by replacing $z \sim \mathcal{N}(\mu, \sigma I)$ with

$$\epsilon \sim \mathcal{N}(0, 1), z = \mu + \sigma \epsilon \quad (2)$$

2.1.1. β -VAE AND FREE BITS

One way of interpreting the ELBO used in the VAE is by considering its two terms $\mathbb{E}[\log p_\theta(x|z)]$ and $\text{KL}(q_\lambda(z|x)||p(z))$ separately. The first term can be thought of as requiring that $p(x|z)$ is high for samples of z from $q_\lambda(z|x)$ - ensuring good reconstruction quality. The second term encourages $q_\lambda(z|x)$ to be close to the prior, which ensures we can generate realistic data by sampling latent codes from $p(z)$. The presence of these two terms suggests a trade-off between the quality of samples and reconstructions - or equivalently, between the rate (amount of information encoded in $q_\lambda(z|x)$) and distortion (data likelihood) (Aleml et al., 2017).

As is, the ELBO has no way of directly controlling this trade-off. A common modification to the ELBO introduces the KL weight hyperparameter β (Bowman et al., 2016; Higgins et al., 2017) producing

$$\mathbb{E}[\log p_\theta(x|z)] - \beta \text{KL}(q_\lambda(z|x)||p(z)) \quad (3)$$

By setting $\beta < 1$, we can encourage the model to weigh reconstruction quality over learning a compact representation.

Another approach for adjusting this trade-off is to only enforce the KL regularization term once it exceeds a threshold (Kingma et al., 2016):

$$\mathbb{E}[\log p_\theta(x|z)] - \max(\text{KL}(q_\lambda(z|x)||p(z)) - \tau, 0) \quad (4)$$

This stems from the interpretation that $\text{KL}(q_\lambda(z|x)||p(z))$ measures the amount of information required to code samples from $p(z)$ using $q_\lambda(z|x)$. Utilizing this threshold therefore amounts to giving the model a certain budget of “free bits” to use when learning the approximate posterior. Note that these modified objectives no longer optimize a lower bound on the likelihood, but as is custom we still refer to the resulting models as “Variational Autoencoders.”

2.1.2. LATENT SPACE MANIPULATION

The broad goal of an autoencoder is to learn a compact representation of the data. For creative applications, we have additional uses for the latent space learned by the model. First, given a point in latent space which maps to a realistic datapoint, points near it in latent space should map to datapoints which are semantically similar. By extrapolation, this implies that all points along a continuous curve connecting two points in latent space should be decodable to a series of datapoints which produce a smooth semantic interpolation in data space. Further, this requirement effectively mandates that the latent space is “smooth” and does not contain any

“holes,” i.e., isolated regions of latent space which do not map to realistic datapoints. Second, we additionally desire that the latent space disentangles meaningful semantic groups in the dataset.

Ideally, these requirements should be satisfied by a Variational Autoencoder if the likelihood and KL divergence terms are both sufficiently small on held-out test data. A more practical test of these properties is to interpolate between points in the latent space and test whether the corresponding points in the input space are interpolated in a semantically meaningful way. Concretely, if z_1 and z_2 are the latent vectors corresponding to datapoints x_1 and x_2 , then we can perform linear interpolation in latent space by computing

$$c_\alpha = \alpha z_1 + (1 - \alpha) z_2 \quad (5)$$

for $\alpha \in [0, 1]$. Our goal is satisfied if $p_\theta(x|c_\alpha)$ is a realistic datapoint for all α , $p_\theta(x|c_\alpha)$ transitions in a semantically meaningful way from $p_\theta(x|c_0)$ to $p_\theta(x|c_1)$ as we vary α from 0 to 1, and that $p_\theta(x|c_\alpha)$ is perceptually similar to $p_\theta(x|c_{\alpha+\delta})$ for small δ . Note that because the prior over the latent space of a VAE is a spherical Gaussian, in high-dimensional spaces samples from the prior are practically indistinguishable from samples from the uniform distribution on the unit hypersphere (Huszár, 2017). In practice we therefore use spherical interpolation (White, 2016) instead of eq. (5).

An additional test for whether our autoencoder will be useful in creative applications measures whether it produces reliable “attribute vectors.” Attribute vectors are computed as the average latent vector for a collection of datapoints which share some particular attribute. Typically, attribute vectors are computed for pairs of attributes, e.g., images of people with and without glasses. The model’s ability to discover attributes is then tested by encoding a datapoint with attribute A, subtracting the “attribute A vector” from its latent code, adding the “attribute B vector”, and testing whether the decoded result appears like the original datapoint with attribute B instead of A. In our experiments, we use the above latent space manipulation techniques to demonstrate the power of our proposed model.

2.2. Recurrent VAEs

While a wide variety of neural network structures have been considered for the encoder and decoder network in a VAE, in the present work we are most interested in models with a recurrent encoder and decoder (Bowman et al., 2016). Concretely, the encoder, $q_\lambda(z|x)$, is a recurrent neural network (RNN) which processes the input sequence $x = \{x_1, x_2, \dots, x_T\}$ and produces a sequence of hidden states h_1, h_2, \dots, h_T . The parameters of the distribution over the latent code z is then set as a function of h_T . The decoder, $p_\theta(x|z)$, uses the sampled latent vector z to set

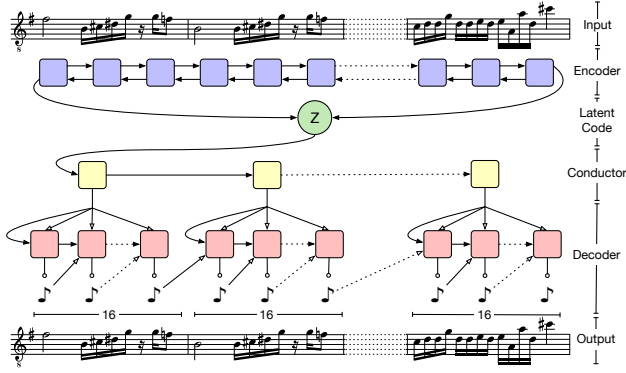


Figure 2. Schematic of our hierarchical recurrent Variational Autoencoder model, “MusicVAE.”

the initial state of a decoder RNN, which autoregressively produces the output sequence $y = \{y_1, y_2, \dots, y_T\}$. The model is trained both to reconstruct the input sequence (i.e., $y_i = x_i, i \in \{1, \dots, T\}$) and to learn an approximate posterior $q_\lambda(z|x)$ close to the prior $p(z)$, as in a standard Variational Autoencoder.

There are two main drawbacks of this approach: first, RNNs are themselves typically used on their own as powerful autoregressive models of sequences. As a result, the decoder in a recurrent VAE is itself sufficiently powerful to produce an effective model of the data, and therefore the decoder can completely ignore the latent code. With the latent code ignored, the KL divergence term of the ELBO can be trivially set to zero, despite the fact that the model is no longer effectively acting as an autoencoder. Second, the model must compress the entire sequence to a single latent vector. While this has been shown to work for short sequences (Bowman et al., 2016; Sutskever et al., 2014), this approach begins to fall apart as the sequence length increases (Bahdanau et al., 2015). In the following section, we present a latent variable autoencoder model which overcomes these issues by using a hierarchical RNN for the decoder.

3. Model

From a high level, our model follows the basic structure used in previously-proposed VAEs for sequential data (Bowman et al., 2016). However, we propose a novel hierarchical decoder, which we show produces substantially better performance on long sequences in section 5. A schematic of our model, which we dub “MusicVAE,” is shown in fig. 2.

3.1. Bidirectional Encoder

For the encoder $q_\lambda(z|x)$, we use a two-layer bidirectional LSTM network (Hochreiter & Schmidhuber, 1997; Schuster & Paliwal, 1997). We process an input sequence $\mathbf{x} = \{x_1, x_2, \dots, x_T\}$ to obtain the final state vectors $\vec{h}_T, \overleftarrow{h}_T$ from the second bidirectional LSTM layer. These

are then concatenated to produce h_T and fed into two fully-connected layers to produce the latent distribution parameters μ and σ :

$$\mu = W_{h\mu}h_T + b_\mu \quad (6)$$

$$\sigma = \log(\exp(W_{h\sigma}h_T + b_\sigma) + 1) \quad (7)$$

where $W_{h\mu}, W_{h\sigma}$ and b_μ, b_σ weight matrices and bias vectors respectively. In our experiments, we use an LSTM state size of 2048 for all layers and a latent dimensionality of 512. As is standard in Variational Autoencoders, μ and σ then parametrize the latent distribution as in eq. (2). The use of a bidirectional recurrent encoder ideally gives the parametrization of the latent distribution longer-term context about the input sequence.

3.2. Hierarchical Decoder

In prior work, the decoder in a recurrent VAE is typically a simple stacked RNN. The decoder RNN uses the latent vector z to set its initial state, and proceeds to generate the output sequence autoregressively. In preliminary experiments (discussed in section 5), we found that using a simple RNN as the decoder resulted in poor sampling and reconstruction for long sequences. We believe this is caused by the vanishing influence of the latent state as the output sequence is generated.

To mitigate this issue, we propose a novel hierarchical RNN for the decoder. Assume that the input sequence (and target output sequence) \mathbf{x} can be segmented into U nonoverlapping subsequences y_u with endpoints i_u so that

$$y_u = \{x_{i_u}, x_{i_u+1}, x_{i_u+2}, \dots, x_{i_{u+1}-1}\} \quad (8)$$

$$\rightarrow \mathbf{x} = \{y_1, y_2, \dots, y_U\} \quad (9)$$

where we define the special case of $i_{U+1} = T$. Then, the latent vector z is passed through a fully-connected layer² followed by a tanh activation to get the initial state of a “conductor” RNN. The conductor RNN produces U embedding vectors $\mathbf{c} = \{c_1, c_2, \dots, c_U\}$, one for each subsequence. In our experiments, we use a two-layer unidirectional LSTM for the conductor with a hidden state size of 1024 and an output dimensionality of 512.

Once the conductor has produced the sequence of embedding vectors \mathbf{c} , each one is individually passed through a shared fully-connected layer followed by a tanh activation to produce initial states for a final bottom-layer decoder RNN. The decoder RNN then autoregressively produces a sequence of distributions over output tokens for each subsequence y_u via a softmax output layer. The embeddings for each subsequence produced by the conductor are also concatenated with the previous output token and passed as

²Throughout, whenever we refer to a “fully-connected layer,” we mean a simple affine transformation as in eq. (6).

input to the decoder RNN for all inputs for a given subsequence. In our experiments, we used a 2-layer LSTM with 1024 units per layer for the decoder RNN.

In principle our use of an autoregressive RNN decoder still allows for the “posterior collapse” problem where the model effectively learns to ignore the latent state. Similiar to (Chen et al., 2017), we find that it is important to limit the scope of the decoder to force it to use the latent code to model long-term structure. For a CNN decoder, this is as simple as reducing the receptive field, but no direct analogy exists for RNNs, which in principle have an unlimited temporal receptive field. To get around this, we reduce the effective scope of the lowest level RNN in the decoder by only allowing it to propagate state within an output subsequence. We initialize each subsequence RNN state with the corresponding embedding passed down by the conductor. This implies that the only way for the decoder to get longer-term context is by using the embeddings produced by the conductor, which in turn depend solely on the latent code. We experimented with an autoregressive version of the conductor where the decoder state was passed back to the conductor at the end of each subsequence, but found it exhibited worse performance. We believe that these combined constraints effectively force the model to utilize the conductor embeddings, and by extension the latent vector, in order to correctly autoencode the sequence.

3.3. Multi-Stream Modeling

Many common sources of sequential data, such as text, consist solely of a single “stream”, i.e., there is only one sequence source which is producing tokens. However, music is often a fundamentally multi-stream signal – a given musical sequence may consist of multiple players producing notes in tandem. Modeling music therefore can also involve modeling the complex inter-stream dependencies. We explore this possibility by introducing a “trio” model, which is identical to our basic MusicVAE except that it produces 3 separate distributions over output tokens - one for each of three instruments (drum, bass, and melody). In our hierarchical decoder model, we consider these separate streams as an orthogonal “dimension” of hierarchy, and use a separate decoder RNN for each instrument. The embeddings from the conductor RNN are used to initialize the state in each layer of each decoder RNN through separate, per-instrument fully-connected layers followed by tanh activations. For our baselines which use a “flat” (non-hierarchical) decoder, we simply use three independent softmax output layers.

4. Related Work

A closely related model is the aforementioned recurrent Variational Autoencoder of (Bowman et al., 2016). Like ours, their model is effectively a Variational Autoencoder

which uses recurrent neural networks for both the encoder and decoder. With careful optimization, (Bowman et al., 2016) demonstrate the ability to generate and interpolate between sentences which have been modeled at the character level. A very similar model was also proposed by (Fabius & van Amersfoort, 2015), which was applied with limited success to music modeling. This approach was also extended to utilize a convolutional encoder and decoder with dilated convolutions in (Yang et al., 2017). The primary difference between these models and ours is the decoder architecture; namely, we use a hierarchical RNN. We use the “flat” RNN decoder as a baseline in section 5 and find it exhibits significantly degraded performance when dealing with very long sequences.

Various additional Variational Autoencoder models with autoregressive decoders have also been proposed. (Semeniuta et al., 2017) consider extensions of the VRAE where the recurrent neural networks are replaced with feed-forward and convolutional networks. The PixelVAE (Gulrajani et al., 2017) marries a Variational Autoencoder with a PixelCNN (van den Oord et al., 2016b) and applies the result to the task of natural image modeling. Similarly, the Variational Lossy Autoencoder (Chen et al., 2017) combine a VAE with a PixelCNN/PixelRNN decoder but also consider limiting the power of the decoder and using a more expressive Inverse Autoregressive Flow (Kingma et al., 2016) prior on the latent codes. Another example of a VAE with a recurrent encoder and decoder is SketchRNN (Ha & Eck, 2018), which successfully models sequences of continuously-valued pen coordinates.

Another similar model to ours is the hierarchical paragraph autoencoder proposed in (Li et al., 2015). They also consider an autoencoder with hierarchical RNNs for the encoder and decoder, where each level in the hierarchy corresponds to natural subsequences in text (e.g., sentences and words). However, they do not impose any constraints on the latent code, and as a result are unable to sample or interpolate between sequences. Our model otherwise differs in its use of a flat bidirectional encoder and lack of autoregressive connections in the first level of the hierarchy.

More broadly, our model can be considered in the sequence-to-sequence framework (Sutskever et al., 2014), which describes models where an encoder produces a compressed representation of an input sequence which is then used to condition a decoder which generates an output sequence. For example, the NSynth model learns embeddings by compressing audio waveforms with a downsampling convolutional encoder and then reconstructing audio with a WaveNet-style decoder (Engel et al., 2017b). Recurrent sequence-to-sequence models are most often applied to sequence *transduction* tasks where the input and output sequences are different. Nevertheless, sequence-to-sequence

autoencoders have been occasionally considered, e.g., as an auxiliary unsupervised training method for semi-supervised learning (Dai & Le, 2015) or in the paragraph autoencoder described above. Again, our approach differs in that we impose structure on the compressed representation (our latent vector) so that we can perform sampling and interpolation.

Finally, there have been many recurrent models proposed where the recurrent states are themselves stochastic latent variables with dependencies across time (Chung et al., 2015; Bayer & Osendorfer, 2014; Fraccaro et al., 2016). A particularly similar example to our model is that of (Serban et al., 2017), which also utilizes a hierarchical encoder and decoder. Their model uses two levels of hierarchy and generates a stochastic latent variable for each sub-sequence of the input sequence. The crucial difference between this class of models and ours is that we use a *single* latent variable to represent the entire sequence, which allows for creative applications such as interpolation and attribute manipulation.

5. Experiments

To demonstrate the power of the MusicVAE, we carried out a series of quantitative and qualitative studies on music data. First, we demonstrate that a simple recurrent VAE like the one described in (Bowman et al., 2016) can effectively generate and interpolate between short sequences of music. Then, we move to significantly longer sequences of music, where our novel hierarchical decoder is necessary in order to effectively model the data. To verify this assertion, we provide quantitative evidence that it is able to reconstruct, interpolate between, and model attributes from data significantly better than the baseline. We conclude with a series of listening studies which demonstrate that our proposed model also produces a dramatic improvement in the perceived quality of samples.

5.1. Data and Training

For our data source, we use MIDI files, which are a widely-used digital score format. MIDI files contain a transcription of the notes played on each individual instrument in a song, as well as meter (timing) information. We collected ≈ 1.5 million unique files from the web, which provided ample data for training our models. We extracted the following types of training data from these MIDI files: 2- and 16-bar melodies (monophonic note sequences), 2- and 16-bar drum patterns (events corresponding to playing different drums), and 16-bar “trio” sequences consisting of separate streams of a melodic line, a bass line, and a drum pattern. For further details on our dataset creation process, refer to appendix A.

We modeled the monophonic melodies and basslines as sequences of sixteenth note events. This resulted in a 130-dimensional output space (categorical distribution over to-

kens) with 128 “note-on” tokens for the 128 possible MIDI pitches, as well tokens for “note-off” and “rest”. For drum patterns, we mapped the 61 drum classes defined by the General MIDI standard (International MIDI Association, 1991) to 9 canonical classes and represented all possible combinations of hits with 512 categorical tokens. For timing, in all cases we quantized notes to sixteenth note intervals, such that each bar consisted of sixteen events. As a result, our two-bar data (used as a proof-of-concept with a “flat” model) resulted in sequences with $T = 32$ and 16-bar data had $T = 256$. For our hierarchical models, we use $U = 16$, meaning each subsequence corresponded to a single bar.

All models were trained using Adam (Kingma & Ba, 2014) with a learning rate annealed from 10^{-3} to 10^{-5} with exponential decay rate 0.9999 and a batch size of 512. The 2- and 16-bar models were run for 50k and 100k gradient updates, respectively. Models were trained using a cross-entropy loss against the ground-truth outputs with teacher forcing.

5.2. Short Sequences

As a proof that modeling musical sequences with a recurrent VAE is possible, we first focused on modeling 2-bar (length-32) monophonic music sequences with a “flat” model. This model was given a tolerance of 48 free bits (≈ 33.3 nats) and had its KL cost weight, β , annealed from 0.0 to 0.2 with exponential rate 0.99999 over 100k steps. We experimented with both monophonic melodic sequences and drum sequences. We found the model was highly accurate at reconstructing its input (table 1, discussed below in section 5.3) and was also able to produce compelling interpolations (fig. 11, appendix) and samples. In other words, it learned to effectively use its latent code without suffering from posterior collapse or exposure bias, as particularly evidenced by the relatively small difference in teacher-forced and sampled reconstruction accuracy (a few percent).

Despite this success, when we applied the model to 16-bar (length-256) sequences, it was unable to reliably model the data. For example, the discrepancy between teacher-forced and sampled reconstruction accuracy increased by more than 27%. This motivated our design of the hierarchical decoder described in section 3.2. In the following sections, we provide an extensive comparison of our proposed model to the “flat” baseline.

5.3. Reconstruction Quality

To begin with, we evaluate whether the hierarchical decoder produces better reconstruction accuracy on 16-bar melodies and drum sequences. For 16-bar models, we give a tolerance of 256 free bits (≈ 177.4 nats) and use $\beta = 0.2$. Table 1 shows the per-step accuracies for reconstructing the sequences in our test set. As mentioned above, for our “flat” baseline, we see signs of posterior collapse, leading to reduc-

Model	Teacher-Forcing		Sampling	
	Flat	Hierarchical	Flat	Hierarchical
2-bar Drum	0.979	-	0.917	-
2-bar Melody	0.986	-	0.951	-
16-bar Melody	0.883	0.919	0.620	0.812
16-bar Drum	0.884	0.928	0.549	0.879
Trio (Melody)	0.796	0.848	0.579	0.753
Trio (Bass)	0.829	0.880	0.565	0.773
Trio (Drums)	0.903	0.912	0.641	0.863

Table 1. Reconstruction accuracies calculated both with teacher-forcing (i.e., next-step prediction) and full sampling. All values are reported on a held-out test set. A softmax temperature of 1.0 was used in all cases, meaning we sampled directly from the logits.

tions in accuracy of $\approx 27 - 32\%$ in accuracy when teacher forcing is removed. Our hierarchical decoder both increases the next-step prediction accuracy and further reduces the exposure bias by better learning to use its latent code. With the hierarchical model, the decrease in sampling accuracy versus teacher forcing only ranges between $\approx 5 - 11\%$. In general, we also find that the reconstruction errors made by our models are reasonable, e.g., notes shortened by a beat or the addition of notes in the appropriate key.

We also explored the performance of our models on our multi-stream “trio” dataset, consisting of 16-bar sequences of melody, bass, and drums. As with single-stream data, the hierarchical model was able to not only achieve much higher accuracy than the flat model but also exhibited a much smaller gap between teacher-forced and sampled performance.

5.4. Interpolations

For creative purposes, we desire interpolations that are smoothly varying and semantically meaningful. In Figure 3, we compare latent-space interpolations from a flat decoder (yellow circles) and hierarchical decoder (red squares) to a baseline of naive blending of the two sequences (green diamonds). We averaged the behavior of interpolating between 1024 16-bar melodies from the evaluation dataset (A) and 1024 other unique evaluation melodies (B), using a softmax temperature of 0.5 to sample the intermediate sequences. We constructed baseline “Data” interpolations by sampling a Bernoulli random variable with parameter α to choose an element from either sequence **a** or **b** for each time step, i.e., $p(x_t = b_t) = \alpha$, $p(x_t = a_t) = 1 - \alpha$.

The top graph of fig. 3 shows that the (sequence length-normalized) Hamming distance, i.e., the proportion of timestep predictions that differ between the interpolation and sequence A, increases monotonically for all methods. The data interpolation varies linearly as expected, following the mean of the Bernoulli distribution. For latent space

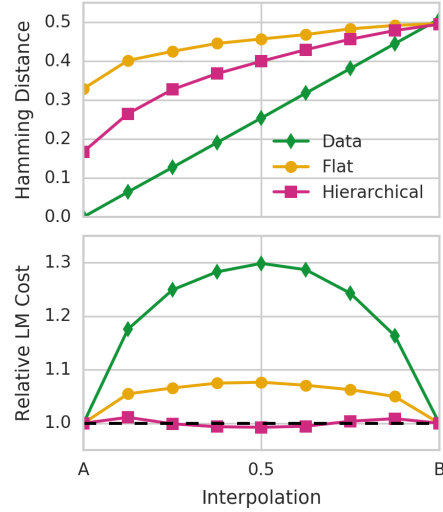


Figure 3. Latent-space interpolation results. All values are averaged over 1024 interpolated sequences. X-axis denotes interpolation between sequence A to B from left to right. *Top*: Sequence-normalized Hamming distance between sequence A and interpolated points sequences. Note the distance from B is symmetric to A (decreasing as A increases) and is not shown. *Bottom*: Relative log probability according to an independently-trained 5-gram language model.

interpolations, the Hamming distances are also monotonic and symmetric, showing that as we interpolate in latent space, the decoded sequences vary smoothly to be less like sequence A and more like sequence B. For example, reconstructions don’t remain on one mode for several steps and then jump suddenly to another mode. Samples have a non-zero Hamming distance at the endpoints because of imperfect reconstructions, and the hierarchical decoder has a lower intercept due to its higher reconstruction accuracy.

For the bottom graph of fig. 3, we first trained a 5-gram language model on the melody dataset (Heafield, 2011). We show the normalized cost for each interpolated sequence given by $C_\alpha / (\alpha C_B + (1 - \alpha) C_A)$, where C_α is the language model cost of an interpolated sequence with interpolation amount α , and C_A and C_B are the costs for the endpoint sequences A and B. The large hump for the data interpolation shows that interpolated sequences in data space are deemed by the language model to be much lower probability than the original melodies. The flat model does better, but produces less coherent interpolated sequences than the hierarchical model, which produces interpolations of equal probability to the originals across the entire range of interpolation.

Figure 1 shows two example melodies and their corresponding midpoint MusicVAE latent interpolation. The interpolation synthesizes semantic elements of the two melodies: playing a similar repeating pattern to A, in a higher register like B, with intermittent sparsity like B, and in a new shared musical key. On the other hand, the data interpola-

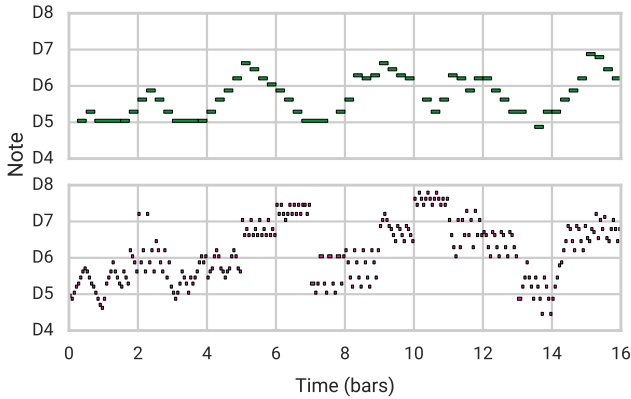


Figure 4. Adding the “high density” attribute latent vector to the latent code of the low-note-density input sequence (top) and decoding the result produces the higher-density sequence on the bottom. See supplemental (Figures 6– 8) for additional examples.

tion just mixes the two and results in harmonic and rhythmic dissonance (supplemental Figure 9).

5.5. Attribute Vector Arithmetic

We can also exploit the structure of the latent space to use “attribute vectors” to alter the attributes of a given sequence. As an example of this, we create an “increased note density” vector by taking the difference of the average embedding of sequences with more than 180 notes and the average embedding of sequences with less than 64 notes. As we can see in Figure 4, adding this vector to an embedding increases the number of notes in the decoded sequence dramatically while preserving many of its original characteristics. Note that notes are not simply repeated - our model also added arpeggios and additional, musically relevant flourishes. Averaged over 1024 sequences with less than 64 notes, adding the attribute vector increases the note density by 398.7%, while the new pitches remain in the original tonal center 99.7% of the time.

5.6. Listening Tests

Capturing whether samples from our model sound realistic is difficult to do with purely quantitative metrics. To compare the perceived sample quality of the different models, we therefore carried out listening test studies for melodies, trio compositions, and drum patterns. Participants were presented with two 16-bar (≈ 30 s) compositions that were either sampled from one of the models or extracted from our evaluation dataset. They were then asked which they thought was more musical on a Likert scale. For each study, 192 ratings were collected, with each source involved in 128 pair-wise comparisons. All samples were generated using a softmax temperature of 0.5.

Figure 5 shows the number of comparisons in which a com-

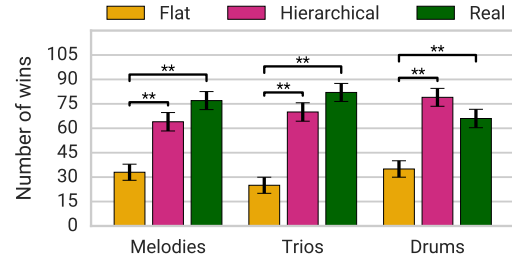


Figure 5. Results of our listening tests. Black bars error indicate estimated standard deviation of means. Double asterisks for a pair indicate a statistically significant difference in ranking.

position from each model was selected as more musical. Our listening test clearly demonstrates the improvement in sample quality gained by using a hierarchical decoder – in all cases the hierarchical model was preferred dramatically more often than the flat model. We did not observe a statistically significant difference between samples from the hierarchical models and real data. For example, the hierarchical drum model was rated higher than the evaluation data, but this difference is not significant. The discrepancy was likely due to a listener bias towards variety, as the true drum data, while more realistic, was also more repetitive and perhaps less engaging.

Further, a Kruskal-Wallis H test of the ratings showed that there was a statistically significant difference between the models: $\chi^2(2) = 37.85, p < 0.001$ for melodies, $\chi^2(2) = 76.62, p < 0.001$ for trios, and $\chi^2(2) = 44.54, p < 0.001$ for drums. A post-hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction showed that participants rated samples from the hierarchical models for melodies, trios, and drums as more musical than samples from their corresponding flat models with $p < 0.01/3$. Participants also ranked real data as more musical than samples from the flat models with $p < 0.01/3$. There was no significant difference between samples from the hierarchical models and real data.

6. Conclusion

We proposed MusicVAE, a recurrent VAE which utilizes a hierarchical decoder for improved modeling of sequences with long-term structure. In experiments on music data, we thoroughly demonstrated through quantitative and qualitative experiments that our model achieves substantially better performance than a “flat” baseline model. In future work, we are interested in testing our model on other types of sequential data. To facilitate future research on recurrent latent variable models, we make our code publicly available.³

³<http://g.co/magenta/musicvae-colab>

7. Acknowledgements

The authors wish to thank David Ha for inspiration and guidance. We thank Erich Elsen for additional editing.

References

- Alemi, Alexander A., Poole, Ben, Fischer, Ian, Dillon, Joshua V., Saurous, Rif A., and Murphy, Kevin. An information-theoretic analysis of deep latent-variable models. *arXiv preprint arXiv:1711.00464*, 2017.
- Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *Third International Conference on Learning Representations*, 2015.
- Bayer, Justin and Osendorfer, Christian. Learning stochastic recurrent networks. In *NIPS Workshop on Advances in Variational Inference*, 2014.
- Bowman, Samuel, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M., Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning*, 2016.
- Carter, Shan and Nielsen, Michael. Using Artificial Intelligence to Augment Human Intelligence. *Distill*, 2017. URL <http://distill.pub/2017/aia>.
- Chen, Xi, Kingma, Diederik P., Salimans, Tim, Duan, Yan, Dhariwal, Prafulla, Schulman, John, Sutskever, Ilya, and Abbeel, Pieter. Variational lossy autoencoder. In *Fifth International Conference on Learning Representations*, 2017.
- Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratharth, Courville, Aaron, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pp. 2980–2988, 2015.
- Dai, Andrew M and Le, Quoc V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, 2015.
- Engel, Jesse, Hoffman, Matthew, and Roberts, Adam. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017a.
- Engel, Jesse, Resnick, Cinjon, Roberts, Adam, Dieleman, Sander, Eck, Douglas, Simonyan, Karen, and Norouzi, Mohammad. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, 2017b.
- Fabius, Otto and van Amersfoort, Joost R. Variational recurrent auto-encoders. In *Third International Conference on Learning Representations*, 2015.
- Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pp. 2199–2207, 2016.
- Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.
- Gulrajani, Ishaan, Kumar, Kundan, Ahmed, Faruk, Taiga, Adrien Ali, Visin, Francesco, Vazquez, David, and Courville, Aaron. PixelVAE: A latent variable model for natural images. In *Fifth International Conference on Learning Representations*, 2017.
- Ha, David and Eck, Douglas. Oph a neural representation of sketch drawings. In *Sixth International Conference on Learning Representations*, 2018.
- Heafield, Kenneth. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 187–197. Association for Computational Linguistics, 2011.
- Higgins, Irina, Matthey, Loic, Pal, Arka, Burgess, Christopher, Glorot, Xavier, Botvinick, Matthew, Mohamed, Shakir, and Lerchner, Alexander. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Fifth International Conference on Learning Representations*, 2017.
- Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Huszár, Ferenc. Gaussian distributions are soap bubbles. *in FERENCE*, 2017.
- International MIDI Association, The. General MIDI level 1 specification. 1991.
- Karras, Tero, Aila, Timo, Laine, Samuli, and Lehtinen, Jaakko. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2014.
- Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. In *Second International Conference on Learning Representations*, 2014.

- Kingma, Diederik P., Salimans, Tim, Jozefowicz, Rafal, Chen, Xi, Sutskever, Ilya, and Welling, Max. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.
- Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, Larochelle, Hugo, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, 2016.
- Li, Jiwei, Luong, Minh-Thang, and Jurafsky, Dan. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*, 2015.
- Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S., and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.
- Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.
- Schuster, Mike and Paliwal, Kuldeep K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- Semeniuta, Stanislau, Severyn, Aliaksei, and Barth, Erhardt. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*, 2017.
- Serban, Iulian Vlad, Sordani, Alessandro, Lowe, Ryan, Charlin, Laurent, Pineau, Joelle, Courville, Aaron, and Bengio, Yoshua. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.
- van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.
- van den Oord, Aaron, Kalchbrenner, Nal, Vinyals, Oriol, Espeholt, Lasse, Graves, Alex, and Kavukcuoglu, Koray. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, 2016b.
- Wei, Xiang, Liu, Zixia, Wang, Liqiang, and Gong, Boqing. Improving the improved training of Wasserstein GANs. In *Sixth International Conference on Learning Representations*, 2018.
- White, Tom. Sampling generative networks: Notes on a few effective techniques. *arXiv preprint arXiv:1609.04468*, 2016.
- Yang, Zichao, Hu, Zhiting, Salakhutdinov, Ruslan, and Berg-Kirkpatrick, Taylor. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, 2017.

A. Dataset Creation Details

The datasets were built by first searching the web for publicly-available MIDI files, resulting in ≈ 1.5 million unique files. We removed those that were identified as having a non- $\frac{4}{4}$ time signature and used the encoded tempo to determine bar boundaries, quantizing to 16 notes per bar (sixteenth notes).

For the 2-bar (16-bar) drum patterns, we used a 2-bar (16-bar) sliding window (with a stride of 1 bar) to extract all unique drum sequences (channel 10) with at most a single bar of consecutive rests, resulting in 3.8 million (11.4 million) examples.

For 2-bar (16-bar) melodies, we used a 2-bar (16-bar) sliding window (with a stride of 1 bar) to extract all unique monophonic sequences with at most a single bar of consecutive rests, resulting in 28.0 million (19.5 million) unique examples.

For the trio data, we used a 16-bar sliding window (with a stride of 1 bar) to extract all unique sequences containing an instrument with a program number in the piano, chromatic percussion, organ, or guitar interval, $[0, 31]$, one in the bass interval, $[32, 39]$, and one that is a drum (channel 10), with at most a single bar of consecutive rests in any instrument. If there were multiple instruments in any of the three categories, we took the cross product to consider all possible combinations. This resulted in 9.4 million examples.

In all cases, we reserve a held-out evaluation set of examples which we use to report reconstruction accuracy, interpolation results, etc.

B. Audio Samples

Synthesized audio for all examples here and in the main text can be found at g.co/magenta/musicvae-samples.

C. Additional Figures and Samples

In the following pages we include additional figures, referenced from the text.

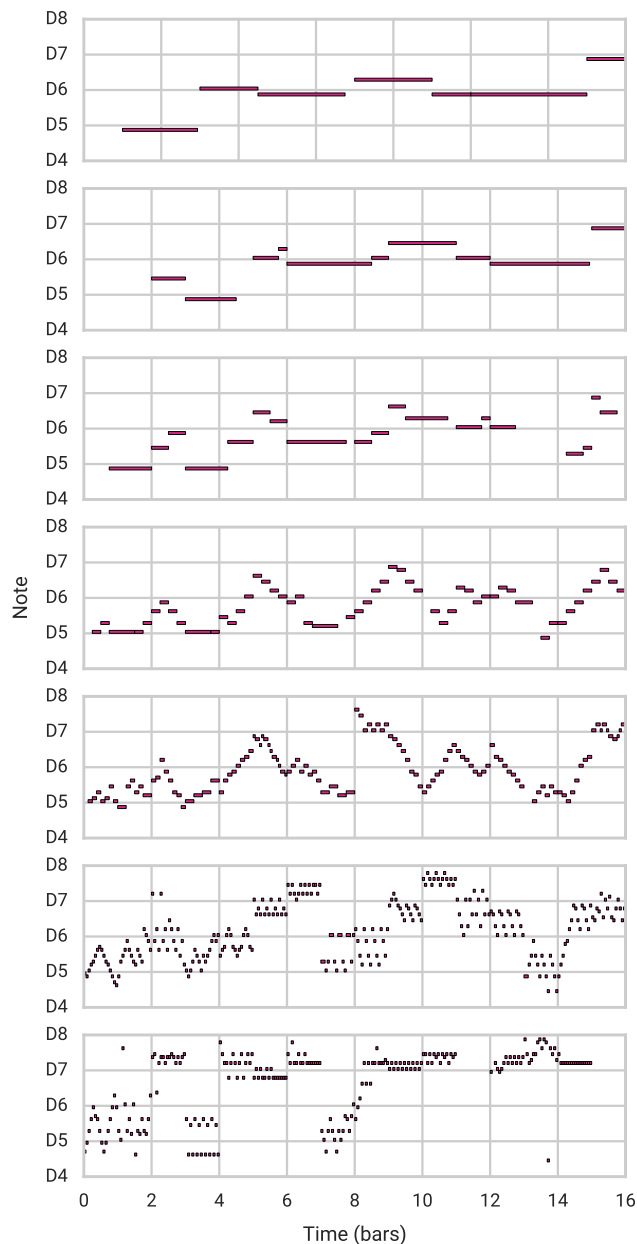


Figure 6. Varying the amount of the “note density” attribute vector, using the example from fig. 4. The amount varies from -1.5 to 1.5 in steps of 0.5, with the center figure corresponding to no attribute vector.

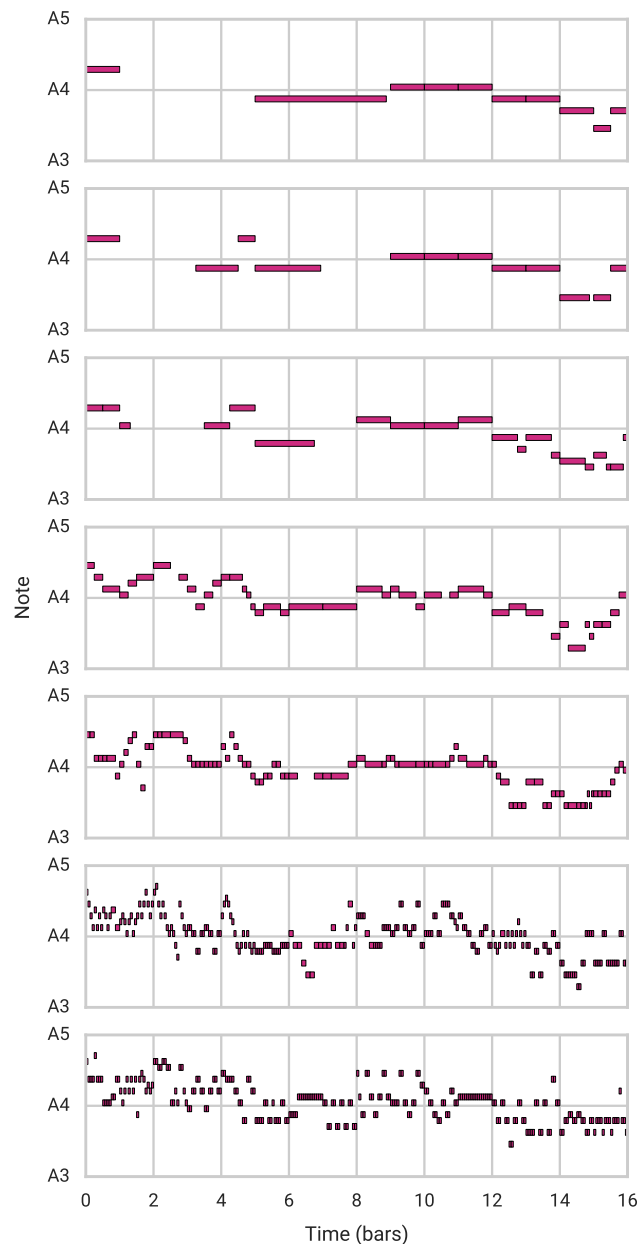


Figure 7. An additional example of increasing note density with attribute vector arithmetic (as in fig. 6). While various properties of the original melodies differ (including pitch range), the effect of the attribute vector is similar.

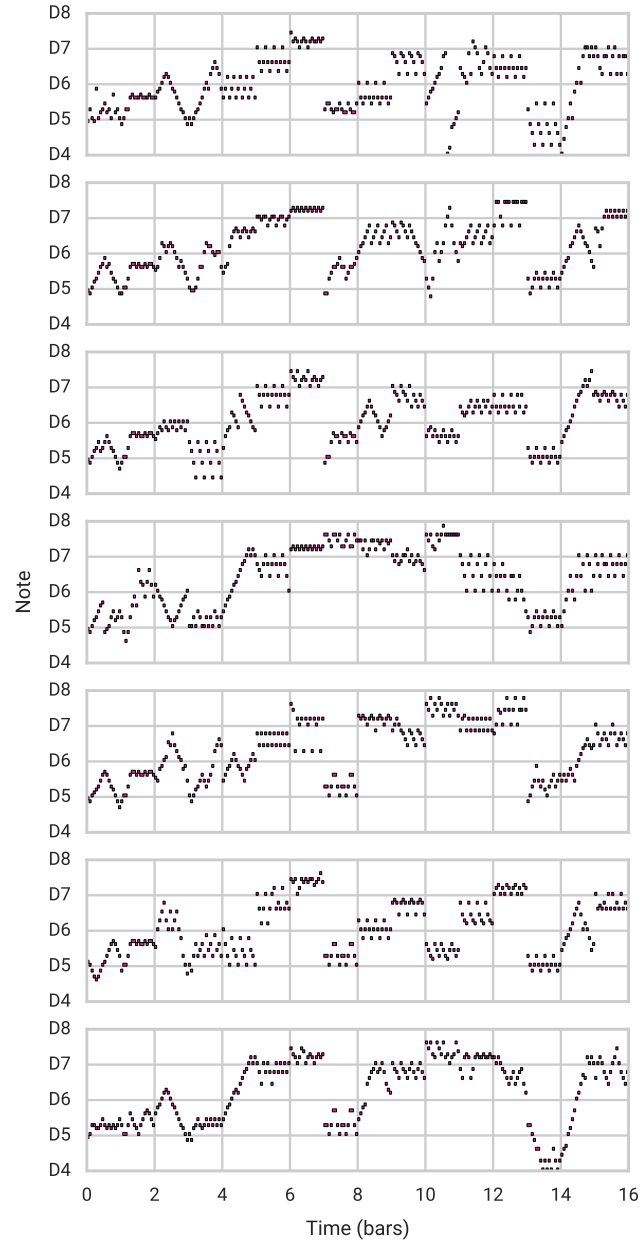


Figure 8. Additional resamplings of the same latent code (corresponding to the second-to-the-bottom in fig. 6). While semantically similar, the specific notes vary due to the sampling in the autoregressive decoder.

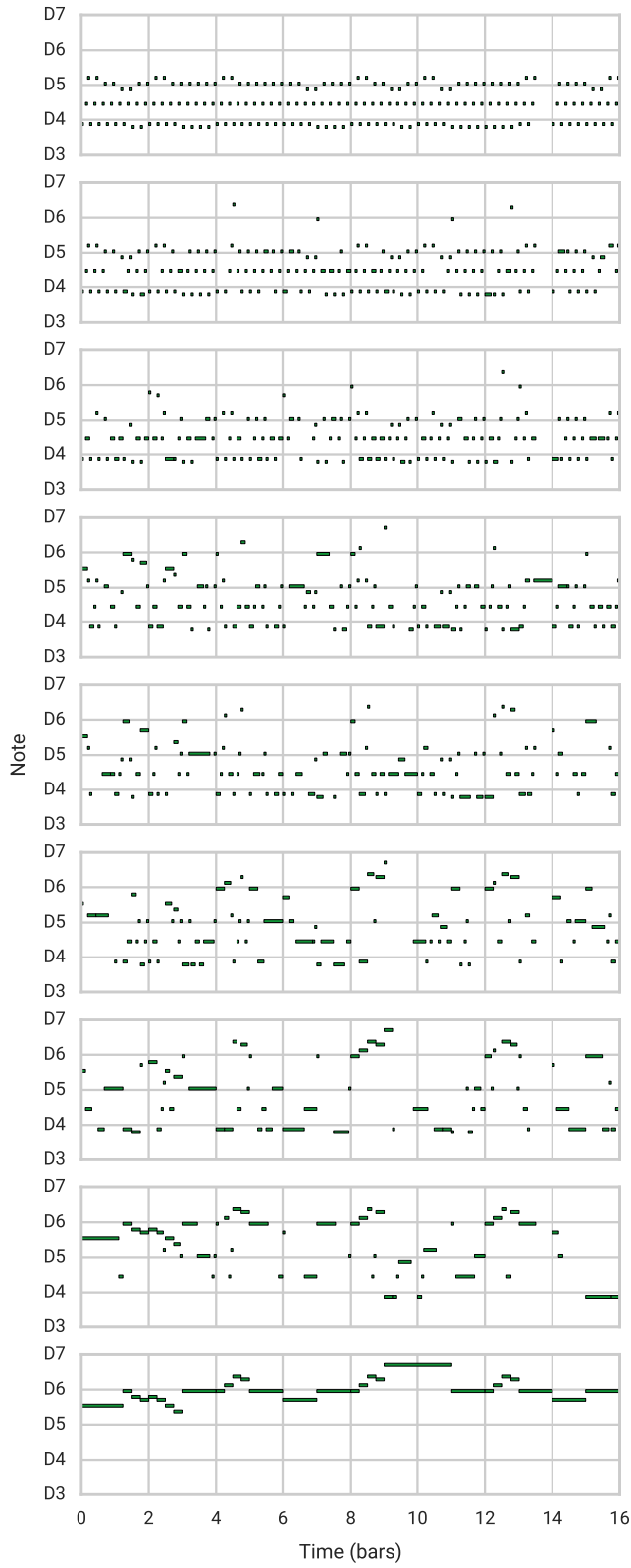


Figure 9. Interpolating between the top and bottom sequence in data space.

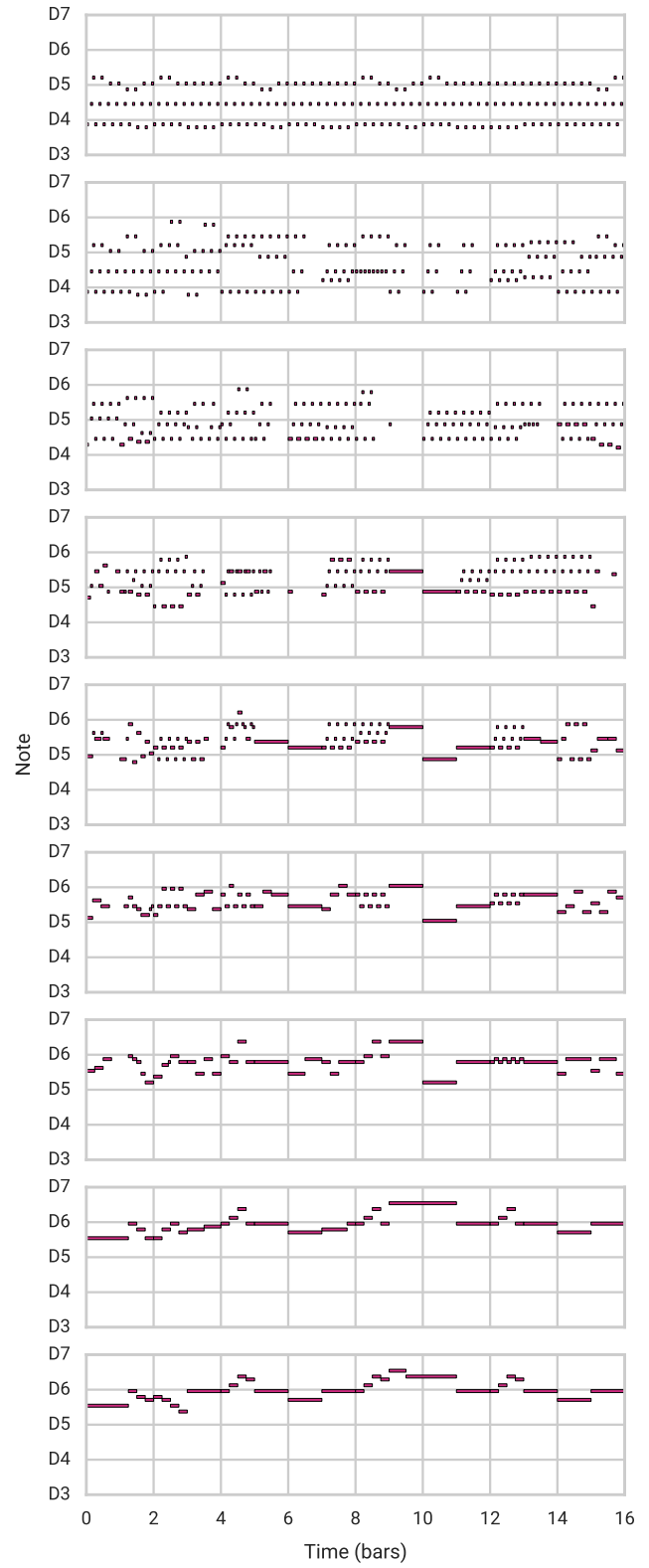


Figure 10. Interpolating between the top and bottom sequence in MusicVAE's latent space.

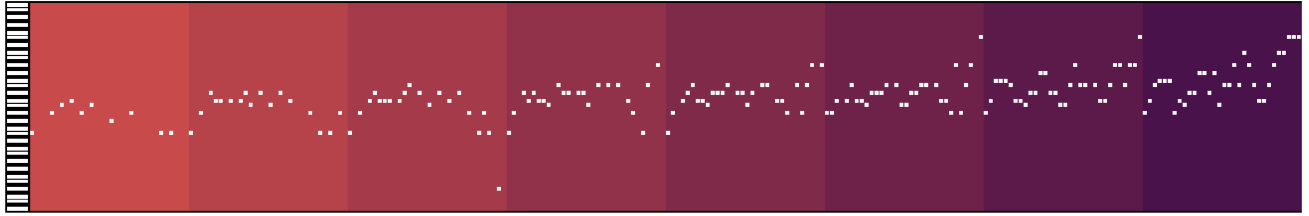


Figure 11. Example 2-bar melody interpolation generated by our “flat” baseline model. Vertical axis is pitch (from A_3 to C_8) and horizontal axis is time. We sampled 6 interpolated sequences between two test-set sequences on the left and right ends. Each 2-bar sample is shown in with a different background color. Audio of an extended, 13-step interpolation between these sequences is available in the supplementary materials.

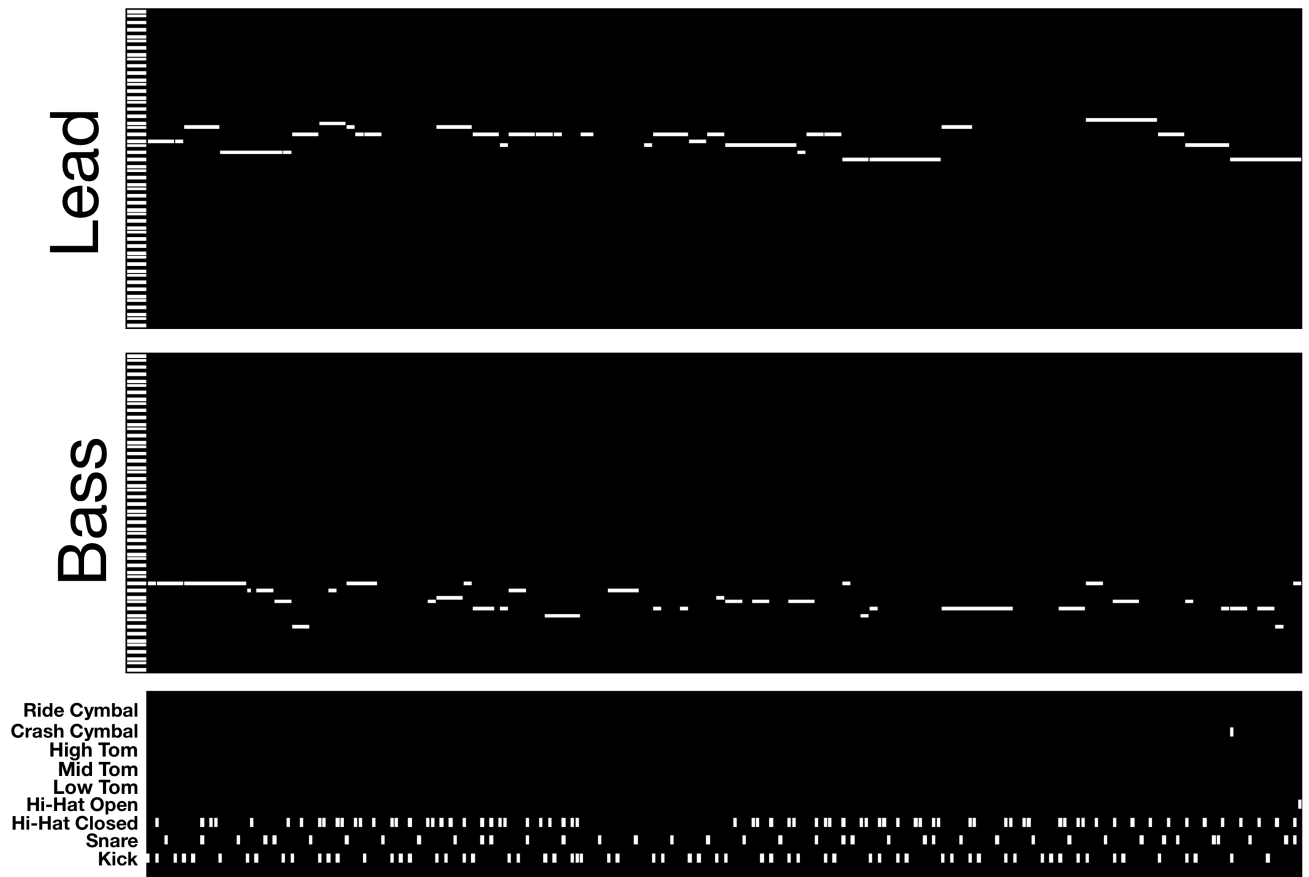


Figure 12. Selected example 16-bar trio sample generated by MusicVAE.