MIDI-VAE: MODELING DYNAMICS AND INSTRUMENTATION OF MUSIC WITH APPLICATIONS TO STYLE TRANSFER

Gino Brunner **Andres Konrad** **Yuyi Wang** **Roger Wattenhofer**
Department of Electrical Engineering and Information Technology
ETH Zurich
Switzerland
brunnegi,konradan,yuwang,wattenhofer@ethz.ch

ABSTRACT

We introduce MIDI-VAE, a neural network model based on Variational Autoencoders that is capable of handling polyphonic music with multiple instrument tracks, as well as modeling the dynamics of music by incorporating note durations and velocities. We show that MIDI-VAE can perform style transfer on symbolic music by automatically changing pitches, dynamics and instruments of a music piece from, e.g., a Classical to a Jazz style. We evaluate the efficacy of the style transfer by training separate style validation classifiers. Our model can also interpolate between short pieces of music, produce medleys and create mixtures of entire songs. The interpolations smoothly change pitches, dynamics and instrumentation to create a harmonic bridge between two music pieces. To the best of our knowledge, this work represents the first successful attempt at applying neural style transfer to complete musical compositions.

1. INTRODUCTION

Deep generative models do not just allow us to generate new data, but also to change properties of existing data in principled ways, and even transfer properties between data samples. Have you ever wanted to be able to create paintings like Van Gogh or Monet? No problem! Just take a picture with your phone, run it through a neural network, and out comes your personal masterpiece. Being able to generate new data samples and perform style transfer requires models to obtain a deep understanding of the data. Thus, advancing the state-of-the-art in deep generative models and neural style transfer is not just important for transforming horses into zebras, [1] but lies at the very core of Deep (Representation) Learning research [2].

While neural style transfer has produced astonishing results especially in the visual domain [21, 37], the progress for sequential data, and in particular music, has been slower. We can already transfer sentiment between restaurant reviews [30, 36], or even change the instrument with which a melody is played [33], but we have no way of knowing how our favorite pop song would have sounded if it were written by a composer who lived in the classical epoch or how a group of jazz musicians would play the Overture of Mozart's Don Giovanni. In this work we take a step towards this ambitious goal. To the best of our knowledge, this paper presents the first successful application of unaligned style transfer to musical compositions. Our proposed model architecture consists of parallel Variational Autoencoders (VAE) with a shared latent space and an additional style classifier. The style classifier forces the model to encode style information in the shared latent space, which then allows us to manipulate existing songs, and effectively change their style, e.g., from Classic to Jazz. Our model is capable of producing harmonic polyphonic music with multiple instruments. It also learns the dynamics of music by incorporating note durations and velocities.

2. RELATED WORK

Gatys et al. [14] introduce the concept of neural style transfer and show that pre-trained CNNs can be used to merge the style and content of two images. Since then, more powerful approaches have been developed [21,37]; these allow, for example, to render an image taken in summer to look like it was shot in winter. For sequential data, autoencoder based methods [30, 36] have been proposed to change the sentiment or content of sentences. Van den Oord et al. [33] introduce a VAE model with discrete latent space that is able to perform speaker voice transfer on raw audio data. Mor et al. [26] develop a system based on a WaveNet autoencoder [12] that can translate music across instruments, genres and styles, and even create music from whistling. Malik et al. [23] train a model to add note velocities (loudness) to sheet music, resulting in more realistic sounding playback. Their model is trained in a supervised manner, with the target being a human-like performance of a music piece in MIDI format, and the input being the same piece but with all note velocities set to the same value. While their model can indeed play music in a more human-like manner, it can only change note velocities, and does not

[1] https://junyanz.github.io/CycleGAN/

learn the characteristics of different musical styles/genres. Our model is trained on unaligned songs from different musical styles. Our model can not only change the dynamics of a music piece from one style to another, but also automatically adapt the instrumentation and even the note pitches themselves. Apart from style transfer, our model can also generate short pieces of music, medleys, interpolations and song mixtures. At the core of our model thus lies the capability to produce music. In the following we will therefore discuss related work in the domains of symbolic and raw audio generation. For a more comprehensive overview we refer the interested readers to these surveys: [4, 13, 16].

People have been trying to compose music with the help of computers for decades. One of the most famous early examples is "Experiments in Musical Intelligence" [9], a semi-automatic system based on Markov models that is able to create music in the style of a certain composer. Soon after, the first attempts at music composition with artificial neural networks were made. Most notably, Todd [31], Mozer [27] and Eck et al. [11] all used Recurrent Neural Networks (RNN). More recently, Boulanger-Lewandowski et al. [3] combined long short term memory networks (LSTMs) and Restricted Boltzmann Machines to simultaneously model the temporal structure of music, as well as the harmony between notes that are played at the same time, thus being capable of generating polyphonic music. Chu et al. [7] use domain knowledge to model a hierarchical RNN architecture that produces multi-track polyphonic music. Brunner et al. [5] combine a hierarchical LSTM model with learned chord embeddings that form the Circle of Fifths, showing that even simple LSTMs are capable of learning music theory concepts from data. Hadjeres et al. [15] introduce an LSTM-based system that can harmonize melodies by composing accompanying voices in the style of Bach Chorales, which is considered a very difficult task even for professionals. Johnson et al. [18] use parallel LSTMs with shared weights to achieve transposition-invariance (similar to the translation-invariance of CNNs). Chuan et al. [8] investigate the use of an image-based Tonnetz representation of music, and apply a hybrid LSTM/CNN model to music generation.

Generative models such as the Variational Autoencoder (VAE) and Generative Adversarial Networks (GANs) have been increasingly successful at modeling music. Roberts et al. introduce MusicVAE [29], a hierarchical VAE model that can capture long-term structure in polyphonic music and exhibits high interpolation and reconstruction performance. GANs, while very powerful, are notoriously difficult to train and have generally not been applied to sequential data. However, Mogren [25], Yang et al. [34] and Dong et al. [10] have recently shown the efficacy of CNN-based GANs for music composition. Yu et al. [35] were the first to successfully apply RNN-based GANs to music by incorporating reinforcement learning techniques.

Researchers have also worked on generating raw audio waves. Van den Oord et al. [32] introduce WaveNet, a CNN-based model for the conditional generation of speech. The authors also show that it can be used to generate pleasing sounding piano music. More recently, Engel et al. [12] incorporated WaveNet into an Autoencoder structure to generate musical notes and different instrument sounds. Mehri et al. [24] developed SampleRNN, an RNN-based model for unconditional generation of raw audio. While these models are impressive, the domain of raw audio is very high dimensional and it is much more difficult to generate pleasing sounding music. Thus most existing work on music generation uses symbolic music representations (see e.g., [3,5,7–10,15,18,23,25,27,29,31,34,35]).

3. MODEL ARCHITECTURE

Our model is based on the Variational Autoencoder [20] (VAE) and operates on a symbolic music representation that is extracted from MIDI [1] files. We extend the standard piano roll representation of note pitches with velocity and instrument rolls, modeling the most important information contained in MIDI files. Thus, we term our model MIDI-VAE. MIDI-VAE uses separate recurrent encoder/decoder pairs that share a latent space. A style classifier is attached to parts of the latent space to make sure the encoder learns a compact latent style label that we can then use to perform style transfer. The architecture of MIDI-VAE is shown in Figure 1, and will be explained in more detail in the following.

3.1 Symbolic Music Representation

We use music files in the MIDI format, which is a symbolic representation of music that resembles sheet music. MIDI files have multiple tracks. Tracks can either be *on* with a certain pitch and velocity, *held* over multiple time steps or be *silent*. Additionally, an instrument is assigned to each track. To feed the note pitches into the model we represent them as a tensor $P \in \{0,1\}^{n_P \cdot n_B \cdot n_T}$ (commonly known as piano roll and henceforth referred to as pitch roll), where n_P is the number of possible pitch values, n_B is the number of beats and n_T is the number of tracks. Thus, each song in the dataset is split into pieces of length n_B. We choose n_B such that each piece corresponds to one bar. We include a "silent" note pitch to indicate when no note is played at a time step. The note velocities are encoded as tensor $V \in [0,1]^{n_P \cdot n_B \cdot n_T}$ (velocity roll). Velocity values between 0.5 and 1 signify a note being played for the first time, whereas a value below 0.5 means that either no note is being played, or that the note from the last time step is being held. The note velocity range defined by MIDI (0 to 127) is mapped to the interval $[0.5, 1]$. We model the assignment of instruments to tracks as matrix $I = \{0,1\}^{n_T \cdot n_I}$ (instrument roll), where n_I is the number of possible instruments. The instrument assignment is a global property and thus remains constant over the duration of one song. Finally, each song in our dataset belongs to a certain style, designated by the style label $S \in \{Classic, Jazz, Pop, Bach, Mozart\}$.

In order to generate harmonic polyphonic music it is important to model the joint probability of simultane-

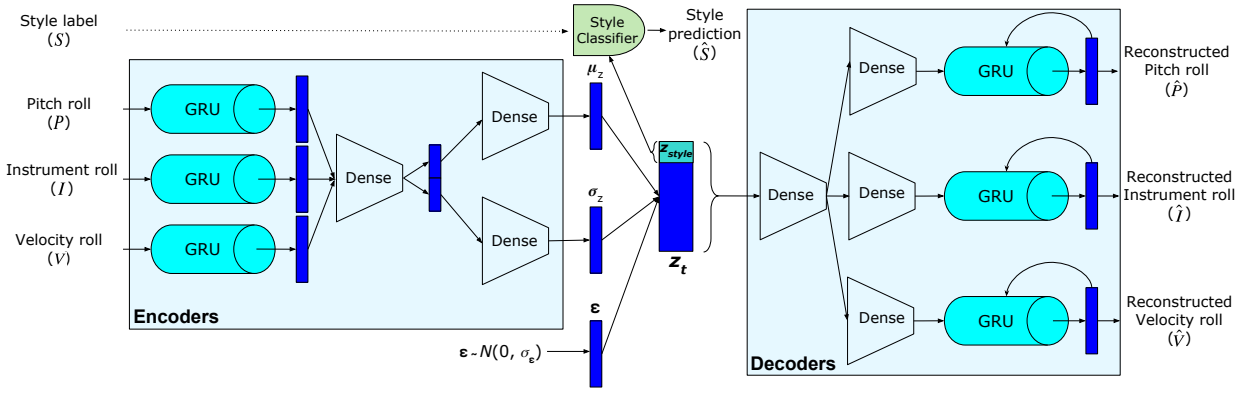

Figure 1. MIDI-VAE architecture. GRU stands for Gated Recurrent Unit [6].

ously played notes. A standard recurrent neural network model already models the joint distribution of the sequence through time. If there are multiple outputs to be produced per time step, a common approach is to sample each output independently. In the case of polyphonic music, this can lead to dissonant and generally "wrong" sounding note combinations. However, by unrolling the piano rolls in time we can let the RNN learn the joint distribution of simultaneous notes as well. Basically, instead of one n_T-hot vector for each beat, we input n_T 1-hot vectors per beat to the RNN. This is a simple but effective way of modeling the joint distribution of notes. The drawback is that the RNN needs to model longer sequences. We use the pretty_midi [28] Python library to extract information from MIDI files and convert them to piano rolls.

3.2 Parallel VAE with Shared Latent Space

MIDI-VAE is based on the standard VAE [20] with a hyperparameter β to weigh the Kullback-Leibler divergence in the loss function (as in [17]). A VAE consists of an encoder $q_\theta(z|x)$, a decoder $p_\phi(x|z)$ and a latent variable z, where q and p are usually implemented as neural networks parameterized by θ and ϕ. In addition to minimizing the standard autoencoder reconstruction loss, VAEs also impose a prior distribution $p(z)$ on the latent variables. Having a known prior distribution enables generation of new latent vectors by sampling from that distribution. Furthermore, the model will only "use" a new dimension, i.e., deviate from the prior distribution, if it significantly lowers the reconstruction error. This encourages disentanglement of latent dimensions and helps learning a compact hidden representation. The VAE loss function is

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\theta(z|x)}[\log p_\phi(x|z)] - \beta D_{KL}[(q_\theta(z|x)||p(z)],$$

where the first term corresponds to the reconstruction loss, and the second term forces the distribution of latent variables to be close to a chosen prior. D_{KL} is the Kullback-Leibler divergence, which gives a measure of how similar two probability distributions are. As is common practice, we use an isotropic Gaussian distribution with unit variance as our prior, i.e., $p(z) = \mathcal{N}(0, I)$. Thus, both $q_\theta(z|x)$ and $p(z)$ are (isotropic) Gaussian distributions and the KL divergence can be computed in closed form.

As described in Section 3.1, we represent multi-track music as a combination of note pitches, note velocities and an assignment of instruments to tracks. In order to generate harmonic multi-track music, we need to model a joint distribution of these input features instead of three marginal distributions. Thus, our model consists of three encoder/decoder pairs with a shared latent space that captures the joint distribution. For each input sample (i.e., a piece of length n_B beats), the pitch, velocity and instrument rolls are passed through their respective encoders, implemented as RNNs. The output of the three encoders is concatenated and passed through several fully connected layers, which then predict $\boldsymbol{\sigma}_z$ and $\boldsymbol{\mu}_z$, the parameters of the approximate posterior $q_\theta(z|x) = \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z)$.[2] Using the reparameterization trick [20], a latent vector $\boldsymbol{z}$ is sampled from this distribution as $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z * \boldsymbol{\epsilon})$ where $*$ stands for element-wise multiplication. This is necessary because it is generally not possible to backpropagate gradients through a random sampling operation, since it is not differentiable. $\boldsymbol{\epsilon}$ is sampled from an isotropic Gaussian distribution $\mathcal{N}(0, \sigma_\epsilon * I)$, where we treat σ_ϵ as a hyperparameter (see Section 4.2 for more details). This shared latent vector is then fed into three parallel fully connected layers, from which the three decoders try to reconstruct the pitch, velocity and instrument rolls. The note pitch and instrument decoders are trained with cross entropy losses, whereas for the velocity decoder we use MSE.

3.3 Style Classifier

Having a disentangled latent space might enable some control over the style of a song. If for example one dimension in the latent space encodes the dynamics of the music, then we could easily change an existing piece by only varying this dimension. Choosing a high value for β (the weight of the KL term in the VAE loss function) has been shown to increase disentanglement of the latent space in the visual domain [17]. However, increasing β has a negative effect

[2] We use notation $\boldsymbol{\sigma}$ for both a variance vector and the corresponding diagonal variance matrix.

Dataset	#Songs	#Bars	Artists
Classic	477	60523	Beethoven, Clementi, ...
Jazz	554	72190	Sinatra, Coltrane, ...
Pop	659	65697	ABBA, Bruno Mars, ...
Bach	156	16213	Bach
Mozart	143	17198	Mozart

Table 1. Properties of our dataset.

on the reconstruction performance. Therefore, we introduce additional structure into the latent space by attaching a softmax style classifier to the top k dimensions of the latent space (z_{style}), where k equals the number of different styles in our dataset. This forces the encoder to write a "latent style label" into the latent space. Using only k dimensions and a weak classifier encourages the encoder to learn a compact encoding of the style. In order to change a song's style from S_i to S_j, we pass the song through the encoder to get $\boldsymbol{z}$, swap the values of dimensions z^i_{style} and z^j_{style}, and pass the modified latent vector through the decoder. As style we choose the music genre (e.g., Jazz, Pop or Classic) or individual composers (Bach or Mozart).

3.4 Full Loss Function

Putting all parts together, we get the full loss function of our model as

$$\begin{aligned}\mathcal{L}_{tot} =& \lambda_P H(P,\hat{P}) + \lambda_I H(I,\hat{I}) \\ &+\lambda_V MSE(V,\hat{V}) + \lambda_S H(S,\hat{S}) - \beta D_{KL}(q||p),\end{aligned} \tag{1}$$

where $H(\cdot,\cdot)$, $MSE(\cdot,\cdot)$ and $D_{KL}(\cdot||\cdot)$ stand for cross entropy, mean squared error and KL divergence respectively. The hats denote the predicted/reconstructed values. The weights λ and β can be used to balance the individual terms of the loss functions.

4. IMPLEMENTATION

In this section we describe our dataset and pre-processing steps. We also give some insight into the training of our model and justification for hyperparameter choices.

4.1 Dataset and Pre-Processing

Our dataset contains songs from the genres Classic, Jazz and Pop. The songs were gathered from various online sources; [3] a summary of the properties is shown in Table 1. Note that we excluded symphonies from our Classic, Bach and Mozart datasets due to their complexity and high number of simultaneously playing instruments. We use a train/test split of 90/10. Each song in the dataset can contain multiple instrument tracks and each track can have multiple notes played at the same time. Unless stated otherwise, we select $n_T = 4$ instrument tracks from each song by first picking the tracks with the highest number of played notes, and from each track we choose the highest voice, meaning picking the highest notes per time step. If a song has fewer than n_T instrument tracks, we pick additional voices from the tracks until we have n_T voices in total. We exclude drum tracks, since they do not have a pitch value. We choose the 16th note as smallest unit. In the most widely used time signature $\frac{4}{4}$ there are 16 16th notes in a bar. 91% of Jazz and Pop songs in our dataset are in $\frac{4}{4}$, whereas for Classic the fraction is 34%. For songs with time signatures other than $\frac{4}{4}$ we still designate 16 16th notes as one bar. All songs are split into samples of one bar and our model auto-encodes one sample at a time. During training we shuffle the songs for each epoch, but keep the bars of a song in the correct order and do not reset the RNN states between samples. Thus, our model is trained on a proper sequence of bars, instead of being confused by random bar progressions.

There are 128 possible pitches in MIDI. Since very low and high pitches are rare and often do not sound pleasing, we only use $n_P = 60$ pitch values ranging from 24 (C_1) to 84 (C_6).

4.2 Model (Hyper-)Parameters

Our model is generally not sensitive to most hyperparameters. Nevertheless we continuously performed local hyperparameter searches based on good baseline models, only varying one hyperparameter at a time. We use the reconstruction accuracy of the pitch roll decoder as evaluation metric. Using Gated Recurrent Units (GRUs) [6] instead of LSTMs increases performance significantly. Using bidirectional GRUs did not improve the results. The pitch roll encoder/decoder uses two GRU layers, whereas the rest uses only one layer. All GRU state sizes as well as the size of the latent space z are set to 256. We use the ADAM optimizer [19] with an initial learning rate of 0.0002. For most layers in our architecture, we found tanh to work better than sigmoid or rectified linear units. We train on batches of size 256. The loss function weights λ_P, λ_I, λ_V and λ_S were set to 1.0, 1.0, 0.1 and 0.1 respectively. λ_p was set to 1.0 to favor high quality note pitch reconstructions over the rest. λ_V was also set to 1.0 because the MSE magnitude is much smaller than the cross entropy loss values.

During our experiments, we realized that high values of β generally lead to very poor performance. We further found that setting the variance of $\boldsymbol{\epsilon}$ to the value of $\sigma_\epsilon = 1$, as done in all previous work using VAEs, also has a negative effect. Therefore we decided to treat σ_ϵ as a hyperparameter as well. Figure 2 shows the results of the grid search. σ_ϵ is the variance of the distribution from which the $\boldsymbol{\epsilon}$ values for the reparameterization trick are sampled, and is thus usually set to the same value as the variance of the prior. However, especially at the beginning of learning, this introduces a lot of noise that the decoder needs to handle, since the values for $\boldsymbol{\mu}_z$ and $\boldsymbol{\sigma}_z$, output by the encoder, are small compared to $\boldsymbol{\epsilon}$. We found that by reducing σ_ϵ, we can improve the performance of our model significantly, while being able to use higher values for β. An annealing strategy for both β and σ_ϵ might produce better

[3] Pop: `www.midiworld.com` / Jazz: `http://midkar.com/jazz/jazz_01.html` / Classic (including Bach, Mozart): `www.reddit.com/r/WeAreTheMusicMakers/comments/3ajwe4/`

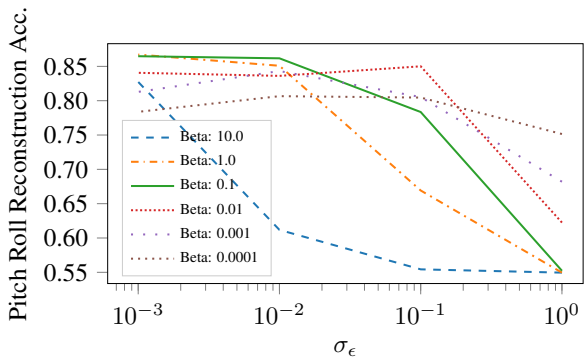

Figure 2. Test reconstruction accuracy of pitch roll for different β and σ_ϵ.

	Pitch		Instrument		Style		Velocity	
	Train	Test	Train	Test	Train	Test	Train	Test
CvJ	0.90	0.85	0.99	0.87	0.98	**0.92**	0.008	**0.029**
CvP	0.96	**0.88**	0.99	**0.89**	0.96	0.91	0.017	0.036
JvP	0.88	0.80	0.99	0.86	0.94	0.69	0.043	0.048
BvM	0.91	0.75	0.99	0.82	0.94	0.74	0.010	0.033

Table 2. Train and test performance of our final models. The velocity column shows MSE loss values, whereas the rest are accuracies.

results, but we did not test this. In the final models we use $\beta = 0.1$ and $\sigma_\epsilon = 0.01$. Note that during generation we sample $\boldsymbol{z}$ from $\mathcal{N}(0, \boldsymbol{\sigma}_{\hat{z}})$, where $\boldsymbol{\sigma}_{\hat{z}}$ is the empirical variance obtained by feeding the entire training dataset through the encoder. The empirical mean $\boldsymbol{\mu}_{\hat{z}}$ is very close to zero.

4.3 Training

All models are trained on single GPUs (GTX 1080) until the pitch roll decoder converges. This corresponds to around 400 epochs, or 48 hours. We train one model for each genre/composer pair to make learning easier. This results in four models that we henceforth call CvJ (trained on Classic and Jazz), CvP (Classic and Pop), JvP (Jazz and Pop) and BvM (Bach and Mozart). The train/test accuracies/losses of all final models are shown in Table 2. The columns correspond to the terms in our model's full loss function (Equation 1).

5. EXPERIMENTAL RESULTS

In this section we evaluate the capabilities of MIDI-VAE. Wherever mentioned, corresponding audio samples can be found on YouTube. [4]

5.1 Style Transfer

To evaluate the effectiveness of MIDI-VAE's style transfer, we train three separate style evaluation classifiers. The input features are the pitch, velocity and instrument rolls respectively. The three style classifiers are also combined to output a voting based ensemble prediction. The accuracy of the classifiers is computed as the fraction of correctly predicted styles per bar in a song. We predict the likelihood of the source style *before* and *after* the style change. If the style transfer works, the predicted likelihood of the source style decreases. The larger the difference, the stronger the effect of the style transfer. Note that for all experiments presented in this paper we set the number of styles $k = 2$, that is, one MIDI-VAE model is trained on two styles, e.g., Classic vs. Jazz. Therefore, the style classifier is binary and a reduction in probability of the source style is equivalent to an increase in probability of the target style of the same magnitude. All style classifiers use two-layer GRUs with a state size of 256. Table 3 shows the performance of MIDI-VAE's style transfer when measured by the ensemble style classifier. We trained a separate MIDI-VAE for each style pair. For each pair of styles we perform a style change on all songs in both directions and average the results. The style transfer works for all models, albeit to varying degrees. In all cases except for JvP, the predictor is even skewed below 0.5, meaning that the target style is now considered more likely than the source style.

	Train Songs			Test Songs		
	Before	After	Diff.	Before	After	Diff.
CvJ	0.92	0.38	**0.54**	0.87	0.39	**0.48**
CvP	0.94	0.43	0.51	0.92	0.45	0.47
JvP	0.72	0.60	0.12	0.72	0.62	0.10
BvM	0.77	0.45	0.32	0.66	0.47	0.19

Table 3. Style transfer performance (ensemble classifier accuracies before and after) between all style pairs.

Table 4 shows the style transfer results measured by each individual style classifier. We can see that pitch and velocity contribute equally to the style change, whereas instrumentation seems to correlate most with the style. For CvJ and CvP, switching the style heavily changes the instrumentation. Figure 3 illustrates how the instruments of all songs in our Jazz test set are changed when switching the style to Classic. Only few instruments are rarely changed (piano, ensemble, reed), whereas most others are mapped to one or multiple different instruments. The instrument switch between genres with highly overlapping instrumentation (JvP, BvM) is much less pronounced. Classifying style based on the note pitches and velocities of one bar is more difficult, as shown by the "before" accuracies in Table 4, which are generally lower than the ones of the instrument roll based classifier. Nevertheless, the style transfer changes pitch and velocity towards the target style. MIDI-VAE retains most of the original melody, while often changing accompanying instruments to suit the target style. This is generally desirable, since we do not want to change the pitches so thoroughly that the original song cannot be recognized anymore. We provide examples of style transfers on a range of songs from our training and test sets on YouTube (see *Style transfer songs*).

[4] https://goo.gl/vb8Yrh

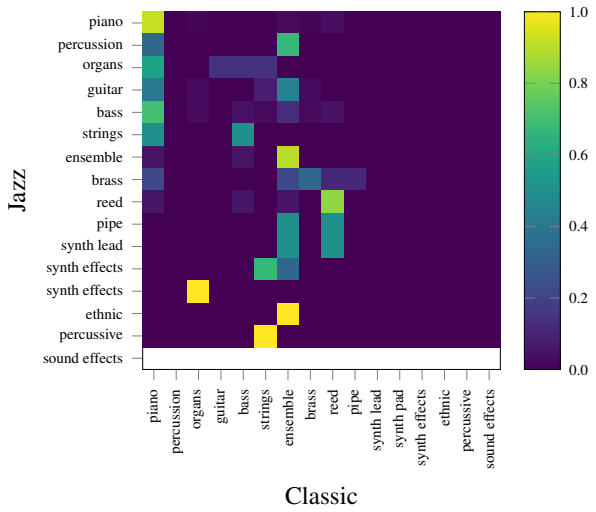

Figure 3. The matrix visualizes how the instruments are changed when switching from Jazz to Classic, averaged over all Jazz songs in the test set.

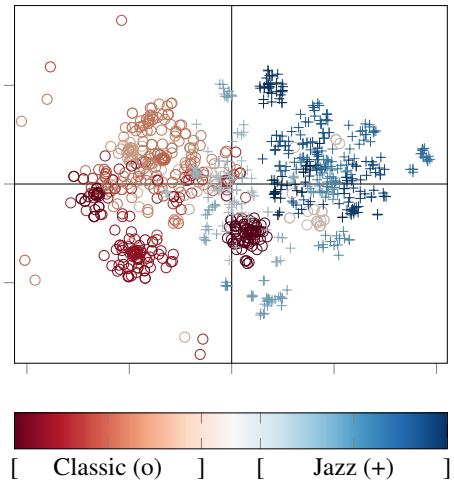

Figure 4. t-SNE plot of latent vectors for bars from 20 Jazz and Classic songs. Bars from the same song were given the same color. Lighter colors mean that the ensemble style classifier was less certain in its prediction.

	Pitch		Velocity		Instrument	
	Bf.	Af.	Bf.	Af.	Bf.	Af.
CvJ Test	0.77	0.66	0.67	0.57	0.90	0.20
CvP Test	0.77	0.67	0.71	0.60	0.91	0.27
JvP Test	0.65	0.63	0.67	0.64	0.67	0.55
BvM Test	0.55	0.47	0.60	0.49	0.64	0.47

Table 4. Average before and after classifier accuracies for all classifiers (pitch/instrument/velocity) for the test set.

5.2 Latent Space Evaluation

Figure 4 shows a t-SNE [22] plot of the latent vectors for all bars of 20 Jazz and 20 Classic pieces. The darker the color, the more "jazzy" or "classical" a song is according to the ensemble style classifier. The genres are well separated, and most songs have all their bars clustered closely together (likely thanks to the instrument roll being constant). Some classical pieces are bleeding over into the Jazz region and vice versa. As can be seen from the light color, the ensemble style classifier did not confidently assign these pieces to either style.

We further perform a sweep over all 256 latent dimensions on randomly sampled bars to check whether changing one dimension has a measurable effect on the generated music. We define 27 metrics, among which are total number of (held) notes, mean/max/min/range of (specific or all) pitches/velocities, and style changes. Besides the obvious dimensions where the style classifier is attached, we find that some dimensions correlate with the total number of notes played in a song, the highest pitch in a bar, or the occurrence of a specific pitch. The changes can be seen when plotting the pitches, but are difficult to hear. Furthermore, the dimensions are very entangled, and changing one dimension has multiple effects. Higher values for $\beta \in \{1, 2, 3\}$ slightly improve the disentanglement of latent dimensions, but strongly reduce reconstruction accuracy (see Figure 2). We added samples to YouTube to show the results of manipulating individual latent variables.

5.3 Generation and Interpolation

MIDI-VAE is capable of producing smooth interpolations between bars. This allows us to generate medleys by connecting short pieces from our dataset. The interpolated bars form a musically consistent bridge between the pieces, meaning that, e.g., pitch ranges and velocities increase when the target bar has higher pitch or velocity values. We can also merge entire songs together by linearly interpolating the latent vectors for two bar progressions, producing interesting mixes that are surprisingly fun to listen to. The original songs can sometimes still be identified in the mixtures, and the resulting music sounds harmonic. We again uploaded several audio samples to YouTube (see *Medleys*, *Interpolations* and *Mixtures*).

6. CONCLUSION

We introduce MIDI-VAE, a simple but effective model for performing style transfer between musical compositions. We show the effectiveness of our method on several different datasets and provide audio examples. Unlike most existing models, MIDI-VAE incorporates both the dynamics (velocity and note durations) and instrumentation of music. In the future we plan to integrate our method into a hierarchical model in order to capture style features over longer time scales and allow the generation of larger pieces of music. To facilitate future research on style transfer for symbolic music, and sequence tasks in general, we make our code and data publicly available. [5]

[5] https://github.com/brunnergino/MIDI-VAE

7. REFERENCES

[1] Midi association, the official midi specifications. https://www.midi.org/specifications. Accessed: 01-06-2018.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[4] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[5] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. JamBot: Music theory aware chord based generation of polyphonic music with LSTMs. In *29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017.

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.

[7] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. *CoRR*, abs/1611.03477, 2016.

[8] Ching-Hua Chuan and Dorien Herremans. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. 2018.

[9] David Cope. Experiments in music intelligence (EMI). In *Proceedings of the 1987 International Computer Music Conference, ICMC 1987, Champaign/Urbana, Illinois, USA, August 23-26, 1987*, 1987.

[10] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[11] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103, 2002.

[12] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1068–1077, 2017.

[13] Jose D. Fernández and Francisco J. Vico. AI methods in algorithmic composition: A comprehensive survey. *J. Artif. Intell. Res.*, 48:513–582, 2013.

[14] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[15] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1362–1371, 2017.

[16] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Comput. Surv.*, 50(5):69:1–69:30, 2017.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[18] Daniel D. Johnson. Generating polyphonic music using tied parallel networks. In *Computational Intelligence in Music, Sound, Art and Design - 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, pages 128–143, 2017.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[21] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 385–395, 2017.

[22] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[23] Iman Malik and Carl Henrik Ek. Neural translation of musical style. *CoRR*, abs/1708.03535, 2017.

[24] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *CoRR*, abs/1612.07837, 2016.

[25] Olof Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, abs/1611.09904, 2016.

[26] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *CoRR*, abs/1805.07848, 2018.

[27] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connect. Sci.*, 6(2-3):247–280, 1994.

[28] Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, pages 84–93, 2014.

[29] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.

[30] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844, 2017.

[31] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

[32] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125, 2016.

[33] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6309–6318, 2017.

[34] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 324–331, 2017.

[35] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858, 2017.

[36] Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders for generating discrete structures. *CoRR*, abs/1706.04223, 2017.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251, 2017.