Full-Band Music Genres Interpolations with Wasserstein Autoencoders

Tijn Borghuis[1,2], **Alessandro Tibo**[3], **Simone Conforti**[4,5,2], **Lorenzo Brusci**[2], **Paolo Frasconi**[6]

[1]Eindhoven University of Technology, [2]Musi-Co, [3]Aalborg University, [4]IRCAM Paris, [5]University of Basel, [6]University of Florence

valentijn.borghuis@musi-co.com, alessandro@cs.aau.dk, simone.conforti@musi-co.com, lorenzo.brusci@musi-co.com, paolo.frasconi@unifi.it

Abstract

We compare different types of autoencoders for generating interpolations between four-instruments musical patterns in the acid jazz, funk, and soul genres. Preliminary empirical results suggest the superiority of Wasserstein autoencoders. The process of generation produces musically sound and creative transitions between different genres and can be of interest to practitioners in the field.

1 Introduction

Automatic music generation is a fast growing area with applications in diverse domains such as gaming, virtual environments, and entertainment industry (see [Pasquier *et al.*, 2017] and references therein). In these days, it intersects another growing area in unsupervised learning, where we are interested in generating (sampling) new patterns from the distribution that produced the dataset.

In previous work [Borghuis *et al.*, 2018], we proposed a generation approach based on the idea of interpolating MIDI drum patterns across different electronic dance music (EDM) genres. The technique is fairly simple and related to approaches in the context of image generation: the autoencoder is trained on a dataset of existing patterns and at prediction time we select a start and a goal pattern (belonging to different genres), compute the corresponding codes generated by the encoder network, interpolate (linearly or spherically) in the embedding space, and use the decoder to produce a sequence of novel patterns that interpolate smoothly from the start to the goal. The approach was successful as measured by human subjects using the creative product analysis matrix (CPAM) approach.

In this work, we deal with the significantly more challenging task of producing genre interpolations for a whole band consisting of four instruments: drums, bass, electric piano, and jazz organ. Genres in our experimentation are Acid Jazz, Funk, and Soul. Human musicians have achieved successful cross-overs between these genres, and the four instruments are integral to all three. Compared to drumming, the present task forces us to deal with the complexity of several musical aspects beyond rhythm such as harmony, melody, and interplay among different instruments. Many researchers have attempted music generation from other angles such as scores described as symbolic sequences [Eck e Schmidhuber, 2002; Boulanger-Lewandow ski *et al.*, 2012; Hadjeres *et al.*, 2017] or wave audio files [van den Oord *et al.*, 2016]. Our approach is more closely related to [Yang *et al.*, 2017] with some major differences: the use of autoencoders instead of GANs (to allow interpolations between two given patterns) and the simultaneous generation of a MIDI score for four musicians, which requires the learner to capture aspect related to their interplay.

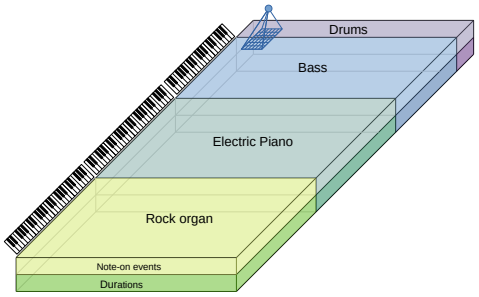

Figure 1: Tensor representation of MIDI patterns

Although this work is still preliminary and in progress, we have obtained some musically interesting results that can be accessed at http://ai.dinfo.unifi.it/Cirox-and-the-Michs/.

2 Materials and Methods

Our dataset consists of 891 patterns composed and played by a professional musician under instructions from the authors. It is approximately balanced across the three genres (361 patterns of Acid Jazz, 255 of Funk and 274 of Soul). Within genres, patterns were constructed in thematic families (sharing musical ideas), taking song structure into account (patterns for verse, chorus, bridge and intro). Each MIDI pattern corresponds to a measure, i.e. four bars (in a 4/4 division) and notes are quantized at 1/32th of a beat, yielding 128 temporal positions. It is represented as a tensor (see Figure 1) where the first axis corresponds to time (size 128), the second axis corresponds to a MIDI note (10 percussive instruments in the drumset, 3 octaves extension for the bass, and four octaves for both the electric piano and the rock organ, yielding a total size of 142), and the third axis is used to distinguish between velocities and durations of MIDI events. In the first channel, a value between 1 and 127 encodes the note velocity on a

note-on event while in the second channel, a value between 1 and 128 encodes the note duration in 32th's. This representation is similar to color image representations (except for the different semantics of channels) and therefore suggests a very natural strategy for learning based on convolutional networks.

We employed three different types of autoencoders, that have been extensively discussed in the vision literature, namely variational (VAE) [Kingma e Welling, 2013], adversarial (AAE) [Makhzani *et al.*, 2015] and Wasserstein (WAE) [Tolstikhin *et al.*, 2017]. While in computer vision it is widely accepted that VAEs tend to produce blurry images, the difference between these kinds of generative models for music has not been evaluated before.

VAEs consist of an encoder which maps an input pattern $x \in \mathbb{R}^m$ to the mean and diagonal covariance $\mu(x), \sigma(x)$ of a distribution $q(z|x)$, and a decoder that takes as input a vector z sampled from a prior distribution $p(z)$ (typically $z \sim \mathcal{N}(0,1)$) in order to obtain $p(x|z)$. VAEs are trained to minimize $KL(q(z)||p(z)) - \log(p(x|z))$, where KL is the Kullback-Leibler divergence.

AAEs are similar to VAEs but are trained to minimize $KL(q(z|x)||p(z)) - \log(p(x|z))$. $KL(q(z|x)||p(z))$ is usually replaced with the adversarial loss which, by introducing a discriminator network Δ, tries to separate *real* latent code vectors generated from the prior $p(z)$ (positive examples) from *fake* latent code vector generated by the encoder (negative examples).

Finally WAEs are a generalization of AAEs, trained to minimize a penalized form of the Wasserstein distance between the the model distribution and the target distribution

$$\inf_{q(z|x)\in\mathcal{Q}} \mathbb{E}_{x\sim p(x)}\mathbb{E}_{z\sim q(z|x)}[c(x,\tilde{x})] + \lambda\, D(q(z),p(z)),$$

where $\tilde{x}$ is the reconstructed pattern associated to x, c is a reconstruction cost function (e.g. $\|x-\tilde{x}\|^2$), $\mathcal{Q}$ is any nonparametric set of probabilistic encoders, D is an arbitrary divergence between $q(z)$ and $p(z)$ (the maximum mean discrepancy in the experiments), and $\lambda > 0$ is a hyperparameter which we set large enough ($\lambda = 2$ in the experiments) to ensure that a well filled latent space.

We designed encoders and decoders of VAE, AAE, and WAE using the same structure, with the only exception for the VAE's encoder output. The encoder consists of four stacked 2D convolutional layers (each followed by batch normalization and rectifiers) with filter size 8×8. Layers have 32, 64, 128, and 256 filters, respectively. The last convolutional layer is linked to a 3-units fully connected layer with linear activation for the AAE and WAE, while to two 3-units fully connected layers with linear activation for the VAE. The decoder consists of a fully connect reshaped layer of (128,138,256) units with rectifiers, three stacked 2D deconvolution layers (each followed by batch normalization and rectifiers, except for the last layer that has a sigmoidal activation function). Filter size was also 8×8 and layers have 128, 64, and 2, filters, respectively. For the reconstruction part of the loss we chose different weights for the velocites (0.05) and the durations (0.25) since the latter appeared more difficult to reconstruct correctly. Finally, we trained on 90% of the available data running 20k epochs of Adam with learning rate 2×10^{-4} (and 5×10^{-4} for the adversarial training).

3 Results and discussion

We trained several variants of the above models before obtaining musically acceptable results. Unlike other areas of machine learning, a major limitation of music generation is the lack of an objective and computable measure of performance of the trained model. Hyperparameter optimization in this context is therefore extremely hard, cannot be automated, and seems to require a trial-and-error approach where the machine learning designer constantly interacts with the musicians to get feedback about the quality of the results. A possible remedy could be to manually label generated patterns according to their quality to enable a supervised assessment, but this is beyond the scope of this work. Although VAEs were shown to be effective for drums only data, here they tend to either produce patterns with limited variability and interplay between instruments (with small codes) or *deconstructed* patterns with many short and low-velocity notes. These effects are much reduced when using AAEs and WAEs but the network size still plays a significant role. We found that wide filters (8x8) and tiny codes (3-dimensional) produce the best results both with AAEs and WAEs. Also, we noted that underfitting (i.e., a large reconstruction error both in training and test patterns) does not necessarily imply a poor musical quality, provided that note-on recall is sufficiently high.

References

[Borghuis *et al.*, 2018] T. Borghuis, A. Tibo, S. Conforti, L. Canciello, L. Brusci, e P. Frasconi. Off the Beaten Track: Using Deep Learning to Interpolate Between Music Genres. *arXiv:1804.09808*, 2018.

[Boulanger-Lewandow ski *et al.*, 2012] N. Boulanger-Lewandow ski, Y. Bengio, e P. Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *ICML*, 2012.

[Eck e Schmidhuber, 2002] D. Eck e J. Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *Proc. IEEE NNSP*, pages 747–756, 2002.

[Hadjeres *et al.*, 2017] G. Hadjeres, F. Pachet, e F. Nielsen. DeepBach: a Steerable Model for Bach Chorales Generation. In *Proc. of ICML'17*, pages 1362–1371, 2017.

[Kingma e Welling, 2013] D.P. Kingma e M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[Makhzani *et al.*, 2015] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, e B. Frey. Adversarial autoencoders. *arXiv:1511.05644*, 2015.

[Pasquier *et al.*, 2017] P. Pasquier, A. Eigenfeldt, O. Bown, e S. Dubnov. An Introduction to Musical Metacreation. *Computers in Entertainment*, 14(2):1–14, 2017.

[Tolstikhin *et al.*, 2017] I. Tolstikhin, O. Bousquet, S. Gelly, e B. Schölkopf. Wasserstein auto-encoders. *arXiv:1711.01558*, 2017.

[van den Oord *et al.*, 2016] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, e K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[Yang *et al.*, 2017] L.C. Yang, S.Y. Chou, e Y.H. Yang. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation. *arXiv:1703.10847*, 2017.