To be published in: *Proceedings of the 27th Conference on Artificial Neural Networks (ICANN)*, Rhodes, Greece, 2018. (Author's Preprint)

Towards End-to-End Raw Audio Music Synthesis

Manfred Eppe, Tayfun Alpay, and Stefan Wermter

Knowledge Technology
Department of Informatics, University of Hamburg
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
{eppe,alpay,wermter}@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de/WTM/

Abstract. In this paper, we address the problem of automated music synthesis using deep neural networks and ask whether neural networks are capable of realizing timing, pitch accuracy and pattern generalization for automated music generation when processing raw audio data. To this end, we present a proof of concept and build a recurrent neural network architecture capable of generalizing appropriate musical raw audio tracks.

Keywords: music synthesis, recurrent neural networks

1 Introduction

Most contemporary music synthesis tools generate symbolic musical representations, such as MIDI messages, Piano Roll, or ABC notation. These representations are later transformed into audio signals by using a synthesizer [16,8,12]. Symbol-based approaches have the advantage of offering relatively small problem spaces compared to approaches that use the raw audio waveform. A problem with symbol-based approaches is, however, that fine nuances in music, such as timbre and microtiming must be explicitly represented as part of the symbolic model. Established standards like MIDI allow only a limited representation which restricts the expressiveness and hence also the producible audio output.

An alternative is to directly process raw audio data for music synthesis. This is independent of any restrictions imposed by the underlying representation, and, therefore, offers a flexible basis for realizing fine tempo changes, differences in timbre even for individual instruments, or for the invention of completely novel sounds. The disadvantage of such approaches is, however, that the representation space is continuous, which makes them prone to generating noise and other inappropriate audio signals.

In this work, we provide a proof of concept towards filling this gap and develop a baseline system to investigate how problematic the large continuous representation space of raw audio music synthesis actually is. We hypothesize that a recurrent network architecture is capable of synthesizing non-trivial musical patterns directly in wave form while maintaining an appropriate quality in terms of pitch, timbre, and timing.

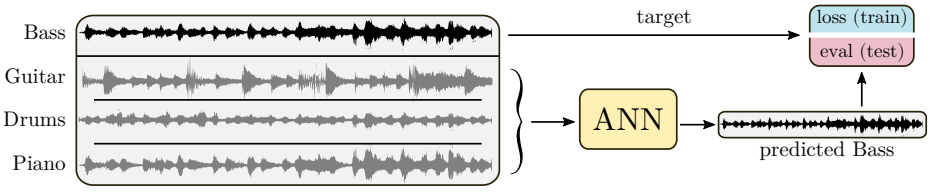


Fig. 1: The practical application and workflow of our system.

The practical context in which we situate our system is depicted in Fig. 1. Our system is supposed to take a specific musical role in an ensemble, such as generating a bassline, lead melody, harmony or rhythm and to automatically generate appropriate audio tracks given the audio signals from the other performers in the ensemble. To achieve this goal, we train a recurrent artificial neural network (ANN) architecture (described in Fig. 2) to learn to synthesize a well-sounding single instrument track that fits an ensemble of multiple other instruments. For example, in the context of a classic rock ensemble, we often find a composition of lead melody, harmony, bass line, and drums. Our proposed system will learn to synthesize one of these tracks, say bass, given the others, i.e., lead melody, harmony and drums. Herein, we do not expect the resulting system to be able to fully replace a human musician, but rather focus on specific measurable aspects. Specifically, we investigate:

1. Timing and beat alignment, i.e., the ability to play a sequence of notes that are temporally aligned correctly to the song's beat.
2. Pitch alignment, i.e., the ability to generate a sequence of notes that is correct in pitch.
3. Pattern generalization and variation, i.e, the ability to learn general musical patterns, such as alternating the root and the 5th in a bass line, and to apply these patterns in previously unheard songs.

We hypothesize that our baseline model offers these capabilities to a fair degree.

2 Related Work

An example for a symbolic approach for music generation, melody invention and harmonization has been presented by Eppe et al. [6,4], who build on *concept blending* to realize the harmonization of common jazz patterns. The work by Liang et al. [12], employs a symbol-based approach with recurrent neural networks (RNNs) to generate music in the style of Bach chorales. The authors demonstrate that their system is capable of generalizing appropriate musical patterns and applying them to previously unheard input. An advanced general artistic framework that also offers symbol-based melody generation is Magenta [16]. Magenta's Performance-RNN module is able to generate complex polyphonic musical patterns. It also supports micro timing and advanced dynamics, but the underlying representation is still symbolic, which implies that the producible audio data is restricted. For example, novel timbre nuances cannot be

generated from scratch. As another example, consider the work by Hung et al. [8], who demonstrate an end-to-end approach for automated music generation using a MIDI representation and Piano Roll representation.

Contemporary approaches for raw audio generation usually lack the generalization capability for higher-level musical patterns. For example, the Magenta framework also involves NSynth [3], a neural synthesizer tool focusing on high timbre quality of individual notes of various instruments. The NSynth framework itself is, however, not capable of generating sequences of notes, i.e., melodies or harmonies, and the combination with the Performance-RNN Magenta melody generation tool [16] still uses an intermediate symbolic musical representation which restricts the produced audio signal. Audio generation has also been investigated in-depth in the field of speech synthesis. For example, the WaveNet architecture [15] is a general-purpose audio-synthesis tool that has mostly been employed in the speech domain. It has inspired the Tacotron text-to-speech framework which provides expressive results in speech synthesis [18]. To the best of our knowledge, however, WaveNet, or derivatives of it, have not yet been demonstrated to be capable of generalizing higher-level musical patterns in the context of generating a musical track that fits other given tracks. There exist some recent approaches to sound generation operating on raw waveforms without any external knowledge about musical structure, chords or instruments. A simple approach is to perform regression in the frequency domain using RNNs and to use a seed sequence after training to generate novel sequences [14,9]. We are, however, not aware of existing work that has been evaluated with appropriate empirical metrics. In our work, we perform such an evaluation and determine the quality of the produced audio signals in terms of pitch and timing accuracy.

3 A Baseline Neural Model for Raw Audio Synthesis

For this proof of concept we employ a simple baseline core model consisting of two Gated Recurrent Unit (GRU) [2] layers that encode 80 Mel spectra into a dense bottleneck representation and then decode this bottleneck representation back to 80 Mel spectra (see Fig. 2). Similar neural architectures have proven to be very successful for various other audio processing tasks in robotics and signal processing (e.g. [5]), and we have experimented with several alternative architectures using also dropout and convolutional layers but found that these variations did not improve the pitch and timing accuracy significantly. We also performed hyperparameter optimization using a Tree-Parzen estimator [1] to determine the optimal number of layers and number of units in each layer. We found that for most experiments two GRU layers of 128 units each for the encoder and the decoder, and a Dense layer consisting of 80 units as a bottleneck representation produced the best results. The dense bottleneck layer is useful because it forces the neural network to learn a Markovian compressed representation of the input signal, where each generated vector of dense activations is independent of the previous ones. This restricts the signals produced during the testing phase of the system, such that they are close to the signals that the system learned from during the training phase.

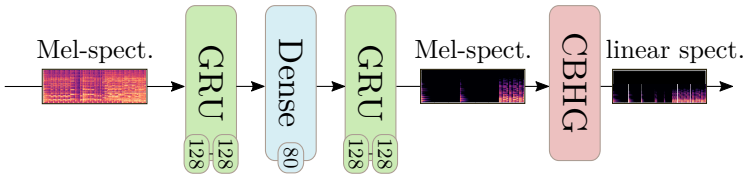


Fig. 2: Our proposed network for mapping the Mel spectra to a dense bottleneck representation, back to Mel spectra, and then to linear frequency spectra.

To transform the Mel spectra generated by the decoding GRU layers back into an audio signal, we combine our model with techniques known from speech synthesis that have been demonstrated to generate high-quality signals [15]. Specifically, instead of using a Griffin-Lim algorithm [7] to transform the Mel spectra into audio signals, we use a CBHG network to transform the 80 Mel coefficients into 1000 linear frequency coefficients, which are then transformed into an audio signal using Griffin-Lim. The CBHG network [11] is composed of a **C**onvolutional filter **B**ank, a **H**ighway layer, and a bidirectional **G**RU. It acts as a sequence transducer with feature learning capabilities. This module has been demonstrated to be very efficient within the Tacotron model for speech recognition [18], in the sense that fewer Mel coefficients, and therefore fewer network parameters, are required to produce high-quality signals [15]. Our loss function is also inspired by the recent work on speech synthesis, specifically the Tacotron [18] architecture: We employ a joint loss function that involves an L1 loss on the Mel coefficients plus a modified L1 loss on the linear frequency spectra where low frequencies are prioritized.

4 Data Generation

To generate the training and testing audio samples, we use a publicly available collection of around 130,000 midi files[1]. The dataset includes various kinds of musical genres including pop, rock, rap, electronic music, and classical music. Each MIDI file consists of several tracks that contain sequences of messages that indicate which notes are played, how hard they are played, and on which channel they are played. Each channel is assigned one or more instruments. A problem with this dataset is that it is only very loosely annotated and very diverse in terms of musical genre, musical complexity, and instrument distribution. We do not expect our proof of concept system to be able to cope with the full diversity of the dataset and, therefore, only select those files that meet the following criteria:

1. They contain between 4 and 6 different channels, and each channel must be assigned exactly one instrument.
2. They are from a similar musical genre. For this work, we select classical pop and rock from the 60s and 70s and select only songs from the following artists: The Beatles, The Kinks, The Beach Boys, Simon and Garfunkel, Johnny Cash, The Rolling Stones, Bob Dylan, Tom Petty, Abba.

[1] https://redd.it/3ajwe4, accessed 18/01/18

3. We eliminate duplicate songs.
4. They contain exactly one channel with the specific instrument to extract.

For this work, we consider bass, reed, and guitar as instruments to extract. The bass channel is representing a rhythm instrument that is present in most of the songs, yielding large amounts of data. The reed channel is often used for lead melody, and guitar tracks often contain chords consisting of three ore more notes. As a result, we obtain 78 songs with an extracted guitar channel, 61 songs with an extracted reed channel, and 128 songs with an extracted bass channel. We split the songs such that 80% are used for training and 20% for testing for each instrument. For each file, we extract the channel with the instrument that we want to synthesize, generate a raw audio (.wav) file from that channel, and chunk the resulting file into sliding windows of 11.5 sec, with a window step size of 6 sec. We then discard those samples which contain a low amplitude audio signal with an average root-mean-square energy of less than 0.2.

5 Results and Evaluation

To obtain results, we trained the system for 40,000 steps with a batch size of 32 samples and generated a separate model for each instrument. For the training, we used an Adam optimizer [10] with an adaptive learning rate. We evaluate the system empirically by developing appropriate metrics for pitch, timing and variation, and we also perform a qualitative evaluation in terms of generalization capabilities of the system. We furthermore present selected samples of the system output and describe qualitatively in how far the system is able to produce high-level musical patterns.

5.1 Empirical Evaluation

For the empirical evaluation, we use a metric that compares the audio signals of a generated track with the original audio track for each song in the test subset of the dataset. The metric considers three factors: timing accuracy, pitch accuracy, and variation.

Timing accuracy. For the evaluation of the timing of a generated track, we compute the onsets of each track and compare them with the beat times obtained from the MIDI data. Onset estimation is realized by locating note onset events by picking peaks in an onset strength envelope [13]. The timing error is estimated as the mean time difference between the detected onsets and the nearest 32nd notes. Results are illustrated in Fig. 3 for bass, guitar and reed track generation. The histograms show that there exists a difference in the timing error between the generated and the original tracks, specifically for the generated bass tracks. Hence, we conclude that the neural architecture is very accurate in timing. This coincides with our subjective impression that we gain from the individual samples depicted in Sec. 5.2. The computed mean error is between 20ms and 40ms, which is the same for the original track. Since the

onset estimation sometimes generates wrong onsets (cf. the double onsets in the original track of *Ob-La-Di, Ob-La-Da*, Sec. 5.2), we hypothesize that the error results from this inaccuracy rather than from inaccurate timing.

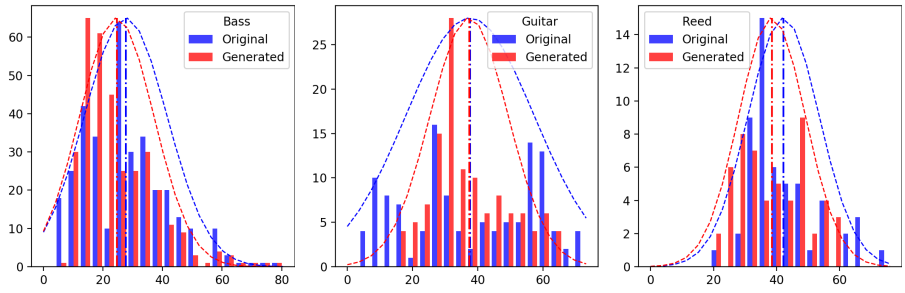


Fig. 3: Timing results for bass, guitar and reed track generation. The x-axis denotes the average error in ms and the y-axis the number of samples in a histogram bin.

Pitch accuracy. We measure the pitch accuracy of the generated audio track by determining the base frequency of consecutive audio frames of 50ms. Determining the base frequency is realized by quadratically interpolated FFT [17], and we compare it to the closest frequency of the 12 semitones in the chromatographic scale over seven octaves. The resulting error is normalized w.r.t. the frequency interval between the two nearest semitones, and averaged over all windows for each audio sample. The results (Fig. 4) show that that the system is relatively accurate in pitch, with a mean error of 11%, 7%, and 5.5% of the frequency interval between the nearest two semitones for bass, guitar, and reed respectively. However, in particular for the bass, this is a significantly larger error than the error of the original track. The samples depicted in Sec. 5.2 confirm these results subjectively, as the produced sound is generally much less clean than the MIDI-generated data, and there are several noisy artifacts and chunks that are clearly outside of the chromatographic frequency spectrum.

Variation. To measure variation appropriateness, we consider the number of tones and the number of different notes in each sample. However, in contrast to pitch and timing, it is not possible to compute an objective error for the amount of variation in a musical piece. Hence, we directly compare the variation in the generated samples with the variation in the original samples and assume implicitly that the original target track has a perfect amount of notes and tones. Hence, to compute the variation appropriateness v we compare the number of original notes (n_{orig}) and tones (t_{orig}) with the number of generated notes (n_{gen}) and tones (t_{gen}), as described in Eq. 1.

$$
\begin{aligned}
v &= v_{\text{notes}} \cdot v_{\text{tones}} \quad \text{with} \\
v_{\text{tones}} &= \begin{cases} \frac{\mathrm{t}_{orig}}{\mathrm{t}_{gen}} & \text{if } \mathrm{t}_{orig} < \mathrm{t}_{gen} \\ \frac{\mathrm{t}_{gen}}{\mathrm{t}_{orig}} & \text{otherwise} \end{cases} \quad v_{\text{notes}} = \begin{cases} \frac{\mathrm{n}_{orig}}{\mathrm{n}_{gen}} & \text{if } \mathrm{n}_{orig} < \mathrm{n}_{gen} \\ \frac{\mathrm{n}_{gen}}{\mathrm{n}_{orig}} & \text{otherwise} \end{cases}
\end{aligned} \tag{1}
$$

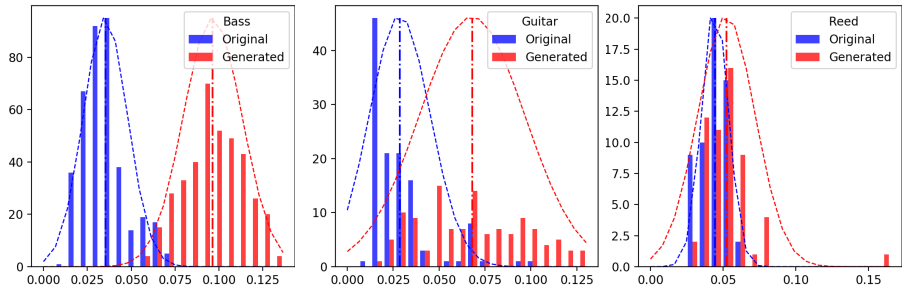


Fig. 4: Pitch accuracy results for bass, guitar and reed track generation; The x-axis denotes the average pitch error in fractions of the half interval between the two closest semitone frequencies.

Results are illustrated in Fig. 5. The histograms show that there are several cases where the system produces the same amount of variation as the original tracks. The average variation value is approximately 0.5 for all instruments. However, we do not consider this value as a strict criterion for the quality of the generated tracks, but rather as an indicator to demonstrate that the system is able to produce tracks that are not too different from the original tracks.

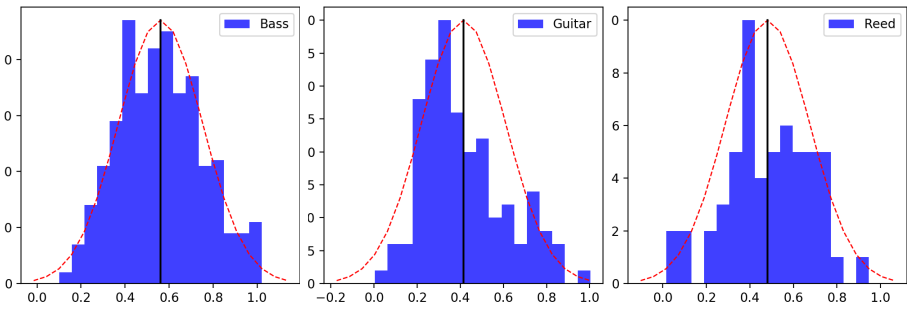


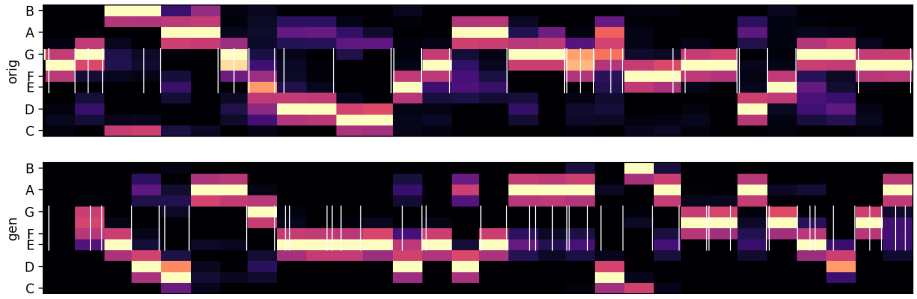
Fig. 5: Variation of generated tracks compared to the original track for three different instruments.

5.2 Qualitative evaluation

To evaluate the generated audio files qualitatively, we investigate the musical patterns of the generated examples. The patterns that we found range from simple sequences of quarter notes over salient accentuations and breaks to common musical patterns like minor and major triads. In the following, analyze two examples of generated bass lines and, to demonstrate how the approach generalizes

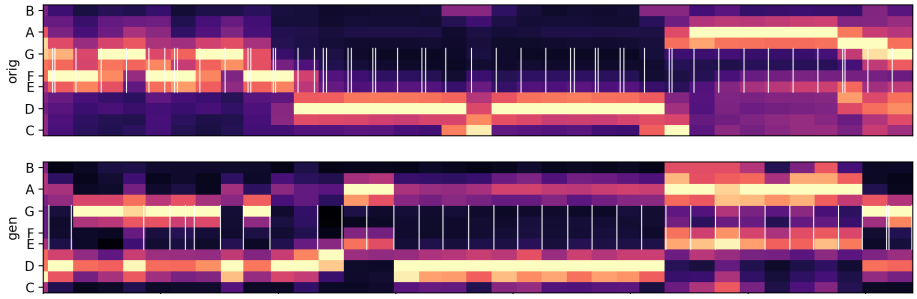
over different instruments, also one example of a generated flute melody. We visualize the samples using beat-synchronous chromagrams with indicated onsets (vertical while lines). The upper chromagrams represent the original melodies and the lower chromagrams the generated ones. Audio samples where the original tracks are replaced by the generated ones are linked with the song titles.

Op. 74 No. 15 Andantino Grazioso - Mauro Giuliano.



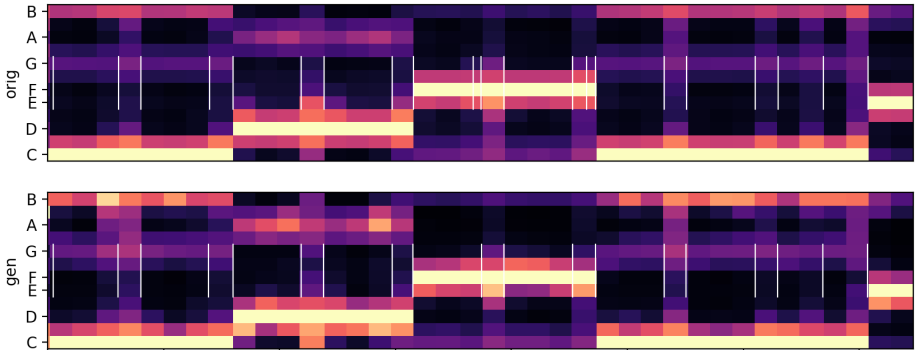
The piece has been written for guitar and flute, and we obtained this result by training the network on all files in our dataset that contain these two instruments. The newly generated flute track differs significantly from the original one although style and timbre are very similar. All notes of the generated track are played in D major scale, same as the original track. The beat is also the same even though the network generates more onsets overall. Near the end of the track, the flute plays a suspended C# which dissolves itself correctly into the tonic chord D. This shows how the network successfully emulates harmonic progression from the original.

The Beatles - Ob-La-Di, Ob-La-Da.



Most generated samples are similar to the illustrated one from *The Beatles - Ob-La-Di, Ob-La-Da*, where the generated notes are in the same key of the original composition, including the timings of chord changes. In some places, however, alternative note sequences have formed as can be seen in the first section of the chromagram, where the F-G is replaced by an D-G pattern, and in the middle section of the chromagram, where the D is exchanged with an A for two beats.

Bob Dylan - Positively 4th Street. In some instances, the generated track contains melodies that are also played by other instruments (e.g. the left hand of the piano often mirrors the bassline). For these cases, we observed that the network has learned to imitate the key tones of other instruments. This results in generated tracks that are nearly identical to the original tracks, as illustrated in the following chromagram of *Positively 4th Street*.



However, while the original bass sequence has been generated by a MIDI synthesizer, the new sample sounds much more bass-like and realistic. This means that our system can effectively be used to synthesize an accurate virtual instrument, which can exploited as a general mechanism to re-synthesize specific tracks.

6 Conclusion

We have presented a neural architecture for raw audio music generation, and we have evaluated the system in terms of pitch, timing, variation, and pattern generalization. The metrics that we applied are sufficiently appropriate to determine whether our base line neural network architecture, or future extensions of it, have the potential to synthesize music directly in wave form, instead of using symbolic representations that restrict the possible outcome. We found that this is indeed the case, as the system is very exact in terms of timing, relatively exact in pitch, and because it generates a similar amount of variation as original music. We also conclude that the system applies appropriate musical standard patterns, such as playing common cadences. Examples like *Positively 4th Street* also show that our system is potentially usable as a synthesizer to enrich and replace MIDI-generated tracks.

As future work, we also want to investigate in how far the system implicitly learns high-level musical features and patterns like cadences and triads, and how it uses such patterns to generate appropriate musical audio data.

Acknowledgments. The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No642667).

References

1. Bergstra, J., Yamins, D., Cox, D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In: International Conference on Machine Learning (ICML) (2013)
2. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In: Neural Information Processing Systems (NIPS) (2014)
3. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., Norouzi, M.: Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. Tech. rep. (4 2017), http://arxiv.org/abs/1704.01279
4. Eppe, M., Confalonieri, R., MacLean, E., Kaliakatsos, M., Cambouropoulos, E., Schorlemmer, M., Kühnberger, K.U.: Computational invention of cadences and chord progressions by conceptual chord-blending. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI). pp. 2445–2451 (2015)
5. Eppe, M., Kerzel, M., Strahl, E.: Deep Neural Object Analysis by Interactive Auditory Exploration with a Humanoid Robot. In: International Conference on Intelligent Robots and Systems (IROS) (2018)
6. Eppe, M., MacLean, E., Confalonieri, R., Kutz, O., Schorlemmer, M., Plaza, E., Kühnberger, K.U.: A Computational Framework for Concept Blending. Artificial Intelligence 256(3), 105–129 (2018)
7. Griffin, D., Jae Lim: Signal estimation from modified short-time Fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing 32(2), 236–243 (4 1984)
8. Huang, A., Wu, R.: Deep Learning for Music. Tech. rep. (2016), https://arxiv.org/pdf/1606.04930.pdf
9. Kalingeri, V., Grandhe, S.: Music Generation Using Deep Learning. Tech. rep. (2016), https://arxiv.org/pdf/1612.04928.pdf
10. Kingma, D.P., Ba, J.L.: Adam: a Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR) (2015)
11. Lee, J., Cho, K., Hofmann, T.: Fully Character-Level Neural Machine Translation without Explicit Segmentation. Transactions of the Association for Computational Linguistics 5, 365–378 (2017)
12. Liang, F., Gotham, M., Johnson, M., Shotton, J.: Automatic Stylistic Composition of Bach Chorales with Deep LSTM. In: Proceedings of the 18th International Society for Music Information Retrieval Conference. pp. 449–456 (2017)
13. Mcfee, B., Raffel, C., Liang, D., Ellis, D.P.W., Mcvicar, M., Battenberg, E., Nieto, O.: librosa: Audio and Music Signal Analysis in Python. In: Python in Science Conference (SciPy) (2015)
14. Nayebi, A., Vitelli, M.: GRUV: Algorithmic Music Generation using Recurrent Neural Networks. Tech. rep., Stanford University (2015)
15. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: WaveNet: A Generative Model for Raw Audio. Tech. rep. (2016), http://arxiv.org/abs/1609.03499
16. Simon, I., Oore, S.: Performance RNN: Generating Music with Expressive Timing and Dynamics (2017), https://magenta.tensorflow.org/performance-rnn
17. Smith, J.O.: Spectral Audio Signal Processing. W3K Publishing (2011)
18. Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al.: Tacotron: Towards End-to-End Speech Synthesis. Tech. rep., Google, Inc. (2017), http://arxiv.org/abs/1703.10135