Automatic Generation of Monophonic Melodies with Complexity Constraints Using Deep Neural Networks

Hany Tawfik, Michael Taenzer, Estefanía Cano, Jakob Abeßer
Semantic Music Technologies Group, Fraunhofer IDMT, Ilmenau, Germany

Introduction

Motivation

Can deep learning be used to generate monophonic melodies with a controllable level of complexity?

- Musicians benefit from sheet music exercises to improve their reading and playing skills.
- These exercises should match and increase their capabilities.
- We use machine learning algorithms to generate practice content matching a musician's skill and style of music, following complexity constraints.
- We focus on generating monophonic melodies in the style of traditional Irish folk music.

Previous work

- Previous research used Markov chains to generate musical sequences, limited to short-term dependencies in the training data [1] [2].
- Recurrent Neural Networks (RNN) have been used [3] [4] to overcome these limitations for polyphonic vocal harmonies, such as the DeepBach system [5].

Proposed Method

Approach

- We investigate two generative network architectures for monophonic melody generation:
- **CharRNN** [6]:
  - character-based model consisting of 3 hidden layers with 512 LSTM blocks each
  - processes melodies in textual ABC notation[1]
  - We use the RMSprop algorithm with batches of 500 samples to reduce training time.
  - The model is trained for 20 epochs.
- **MelodyRNN**, proposed by the Magenta[2] research project:
  - uses 2 LSTM-layers with 128 nodes each
  - generates melodies in MIDI format
  - *Attention*-method gives weights to events that happened in previous time steps. The model thereby decides the degree of attention to pay to past events, enabling it to learn longer-term structure.

Dataset and complexity measure

- Training data stems from [6], originally collected from a weekly repository[3].
- We classify the input dataset into several subsets using **rhythmic** features and an academic standard from the ABRSM[4], which defines musical complexity.
- To ensure smooth classification, we compute a melody's average rhythmic complexity by giving weights to each note in respect to its duration value (see table 1).

complexity	*indicators*
Level 1	half, quarter, 8th
Level 2	whole, dotted half
Level 3	dotted whole, dotted quarter, dotted 8th, 16th
Level 4	6/8th time signature
Level 5	dotted 16th, dotted 32nd, dotted 64th, 32nd, 64th, triplets

Table 1

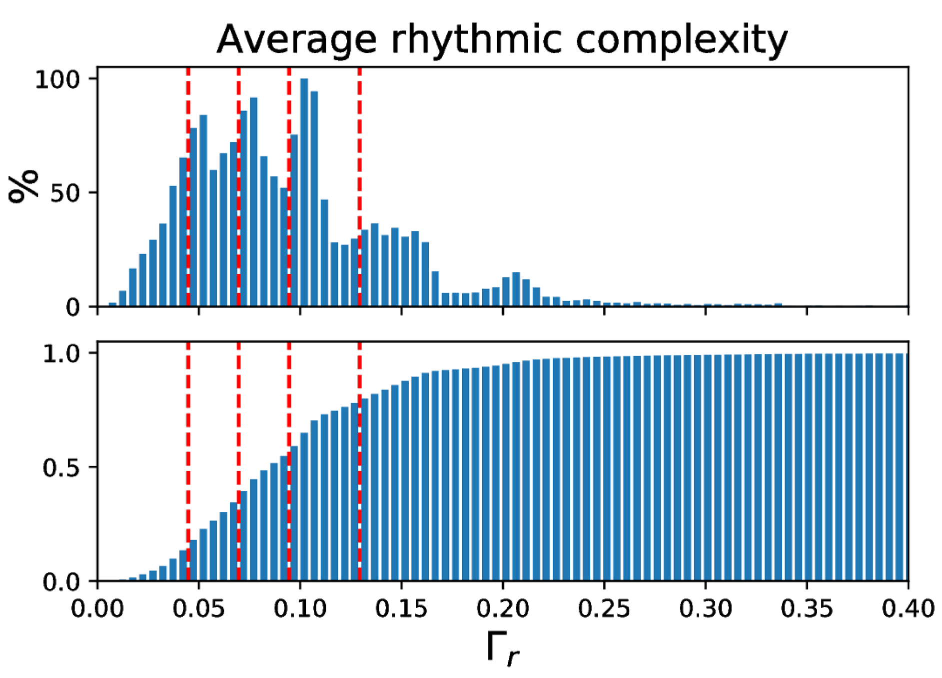

Figure 1

- Figure 1 shows the distribution of rhythmic complexity and the cumulative sum we use to create classes with an approx. equal number of files (red dashed lines).

1 http://abcnotation.com/wiki/abc:standard:v2.1
2 https://github.com/tensorflow/magenta
3 https://thesession.org
4 Associated Board of the Royal Schools of Music. We use the outlines of the violin standard: https://de.abrsm.org/en/our-exams/bowed-strings-exams/violin-exams

System pipeline

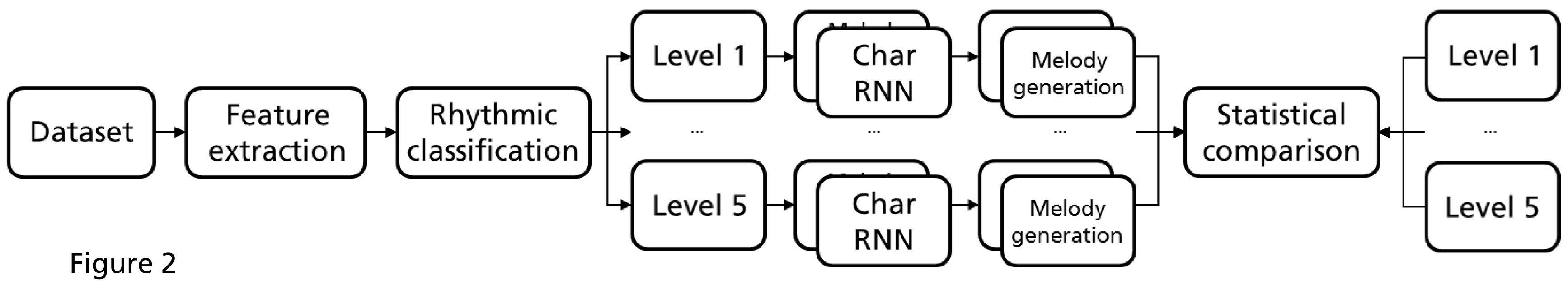

Figure 2

- First, features are extracted from the melodies. These are used to classify the dataset into five subsets of increasing complexity levels.
- Both model architectures are trained on each of the subsets.
- Then, we generate 1000 files per level per model for comparison to the training melodies.
- Our main hypothesis: after training a model with melodies of higher complexity, the model will also generate melodies of higher complexity.

Evaluation & Results

- To test our main hypothesis, we represent each subset of the original and generated dataset by the average rhythmic complexity Γ_r over its melodies.
- Statistical comparisons between Γ_r of the training data subsets and generated data subsets.
- We achieve a significant correlation for the CharRNN model architecture (see table 2).

	Pearson correlation coefficient r	p
CharRNN	0.9844	0.0023
MelodyRNN	0.8035	0.1013

Table 2

- Our results show:
  - Existing generative neural networks for melody generation can be trained to follow complexity constraints in order to match complexity levels.
- We can assume:
  - The generated melodies are fit to support exercising musicians looking for individual practice material.

Acknowledgements

This work has been supported by the German Research Foundation (AB 675/2-1).

References

[1] L. A. Hiller Jr. and L. M. Isaacson. Musical composition with a high speed digital computer. University of Illinois, Urbana, IL, USA, 1957

[2] B. Manaris, D. Hughes, and Y. Vassilandonakis. Monterey mirror: Combining Markov models, genetic algorithms, and power laws. An experiment in interactive evolutionary music performance. In Proceedings of IEEE Congress on Evolutionary Computation, pages 33–40, New Orleans, Louisiana, USA, 2011.

[3] J.-P. Briot, G. Hadjeres, and F. Pachet. Deep learning techniques for music generation - a survey. arXiv preprint arXiv:1709.01620, 2017

[4] N. Jaques, S. Gu, R. E. Turner, and D. Eck. Tuning recurrent neural networks with reinforcement learning. In 5th International Conference on Learning Representations, Toulon, France, 2017.

[5] G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for bach chorales generation. In 34th International Conference on Machine Learning, Sydney, Australia, 2017

[6] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova. Music transcription modeling and composition using deep learning. In 1st Conference on Computer Simulation of Musical Creativity, University of Huddersfield, UK, 2016.

Fraunhofer IDMT