University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Marcin Papierzyński
Student no. 345782

Music Style Transfer

Master's thesis
in COMPUTER SCIENCE

Supervisor:
dr hab. Marek Cygan
Instytut Informatyki

September 2019

Supervisor's statement

Hereby I confirm that the presented thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master of Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the presented thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

TODO

Keywords

style transfer, music, midi, neural networks

Thesis domain (Socrates-Erasmus subject area codes)

11.4 Artificial Intelligence

Subject classification

Applied computing
Arts and humanities
Sound and music computing

Tytuł pracy w języku polskim

Transfer Stylu Muzycznego

Contents

Chapter 1

Introduction

Transfer of style between images is a well-known problem. Currently it can be realized between any two images by using convolutional neural networks. The analogous style transfer done for music is a much less explored topic and it's challenging even to define what exactly a musical style is.

In this paper the goal is to define some sensible notion of musical style and create a model that can create a different arrangement for any song based on style extracted from any other song. The music used for training is assumed to be in MIDI format and can use any range of instruments. The instruments supported by the model must be set during training, but the supported styles do not – we want the trained model to be able to extract style from any song.

Chapter 2

Other works

There exist previous works doing music style transfer to some extent. A few examples are shortly described below. All of them, however, are performing a simpler kind of style transfer.

2.1. Neural Translation of Musical Style

In Neural Translation of Musical Style [1] authors created a model than can learn to perform any given piano piece in either jazz or classical style. The model does this by regression of notes' velocities (loudness), so it's effectively creating an interpretation of a given piano piece. Therefore it's much simpler, because it only supports one instrument (piano) and cannot create a completely different arrangement.

2.2. Symbolic Music Genre Transfer with CycleGAN

In Symbolic Music Genre Transfer with CycleGAN [2] the described model can create new arrangements but it can still only generate piano and the styles need to be set during training (in this case it's jazz, classic and pop).

2.3. Play as You Like: Timbre-enhanced Multi-modal Music Style Transfer

In Timbre-enhanced Multi-modal Music Style Transfer [3] the model can operate on different instruments, but the styles still need to be set during training (the styles are actually instruments, the authors are using guitar, piano and string quartet). Like in the previous examples, it doesn't allow to transfer style between any two songs.

Chapter 3

Results

3.1. Musical style and composition

For musical style transfer, the two main notions concerning songs are composition and style. I start by giving a rough definition of them.

Composition is any information about a song that is related to specific moment in time, e.g. if the music gets louder or some instrument starts playing. The most important part of composition is the melody (played by one or more instruments).

The style, on the other hand, is information not related to any specific moment in time, e.g. the general mood of the song or the way specific instruments are used (for example, which instruments play the main melodic line and which ones create a backing).

Since music is a form of art, even though it can be understood and formally described, it will always be at least partly subjective. This means that the exact way to split songs into composition and style is not unique (for example, used instruments may naturally be considered part of style, but they can also change in time). The main assumption, however, is that composition-style splitting can be performed in *some* fashion which can be learned by a complex enough model.

Since any song can be naturally interpreted as a time sequence, I will use a recurrent neural network as a model for splitting the input song into composition and style. Composition, just like the input, is a time series, but simpler, with less features. The style, on the other hand, is a single vector representing the whole song. During training, the model splits the input song into composition and style, and then combines them to recreate the original song. Because composition is much smaller than the input song, the model must include useful information in the style vector as well.

After training, I can use the model to extract style from some song and combine it with the composition extracted from other song. The hypothesis is that this approach will allow for a sensible music style transfer.

3.2. Experimental results

TODO

Chapter 4

Framework

4.1. Tools

The whole project is implemented in Python. The machine learning framework I use is PyTorch, along with mido – a Python library for working with MIDI files.

4.2. Data

The dataset I use is Lakh MIDI Dataset[1]. It contains over 100,000 songs in MIDI format and covers various genres, including pop, rock or classical.

[1] https://colinraffel.com/projects/lmd/

Chapter 5

Data flow

TODO
3 stages of data flow

5.1. Style extraction

TODO
The input to the model:

- mode: one-hot encoded mode of the song (major or minor)
- instruments: one-hot encoded instruments used in the song
- song: the song content (at least one pitched instrument and optionally percussion)

Model has to split the input song into composition and style. Style is simply a vector. Composition is melody and rhythm, described in more detail below.

Input contains the song content: information which notes are played on each instrument in any moment in the song.

Melody should encode song content but with no information about specific instruments (it must combine them).

Rhythm is like melody but with no information about specific notes.

Melody encodes all pitched instruments used in the song. Rhythm encodes all pitched instruments and percussion, if it's used. Melody and rhythm, along with the style vector, should enable to recreate all the instruments used in the song.

5.2. Predicting song info

TODO
Based on style and rhythm, the model must then predict basic information about a song:

- mode (classification)
- tempo (regression)
- used instruments (classification)

5.3. Style applying

TODO

Given style vector, melody and rhythm, the model must reconstruct the original song.

Chapter 6

Data representation

TODO

MIDI format: sequence of event (like "play that note"). MIDI has 16 channels, each one can be a different instrument.

The shape of the input song is (batch, channel, bar, beat, beat fraction, note, note features)

where 'channel' refers to MIDI channel (an instrument). Dimensions 'bar', 'beat' and 'beat fraction' specify the exact moment in the song when the note is being played.

Note features for pitched are:

- velocity (loudness)
- duration
- accidentals (raise or lower the note one semitone)

For percussion – only velocity and duration.

I use note's span of 8 octaves, each with 12 sounds, which gives 96 possible notes in total.

Typical way of encoding notes is to have each coordinate in a 'note' dimension correspond to specific note (so 'note' dimension would have length 96). However, this poses a problem for style transfer, because different set of notes is used depending on the mode of the song (major or minor). This means that the same song in a different mode would use different sounds in its melody (so melody representation would partially impose style).

To remedy that, I encode the notes relative to the scale that the song is in. So note C in a song in C major would be encoded the same way as note A in a song in A minor. That way changing the scale will not affect melody representation.

This way of encoding notes only allows to encode notes contained in the scale used (7 out of 12 notes in each octave). For example, if the song is in C major, we could only encode "white keys". To allow for encoding all remaining notes as well, each note has additional features (called accidentals) that can raise it or lower it one semitone. That way we can represent all 12 notes in each octave.

The downside of this solution is that I need to recognize the scale of the song, which is in itself not a trivial problem. I use fairly simple heuristics based on the Krumhansl-Schmuckler key-finding algorithm (http://rnhart.net/articles/key-finding/). They are predicting the scale of the song based on the frequency of notes used.

Melody shape: (batch, channel, bar, beat, beat fraction, note, features)

Same as input shape but with no 'instrument' dimension. Also, it can have different number of features.

Rhythm shape: (batch, channel, bar, beat, beat fraction, features)

Same as melody shape, but with no 'note' dimension.

Style shape: (batch, features)

Number of features in the melody should be low enough so that the model cannot simply remember all the instruments in the input. Instead, the model will need to learn some compressed high-level representation of the melody, from which it will be later able to reconstruct the input instruments (using the style vector).

Rhythm contains additional information about the melody (but not explicitly related to specific notes) and features of percussion if it is present in the input song. The main reason for introducing rhythm alongside melody is to be able to represent percussion (the only unpitched instrument) but at the same time do not force it if it's not present in the original song (the model should be able to add percussion to a song that originally didn't have it).

Chapter 7

Model

TODO

Convolutions, LSTM.

The loss function is a combination of various loss functions:

- notes loss (smooth F1 score measuring if the model is playing the right notes)
- accidentals, instruments and mode prediction loss (cross entropy)
- velocity, duration and tempo regression loss (MSE)

Chapter 8

Experiments

TODO

The model in each iteration is given one random song from the dataset. It needs to split it into composition and style, and then reconstruct the original song.

After training, the model is used to extract style and composition from two songs and then generate songs from these compositions but with switched styles.

Bibliography

[1] Iman Malik, Carl Henrik Ek, *Neural Translation of Musical Style*, https://arxiv.org/abs/1708.03535.

[2] Gino Brunner, Yuyi Wang, Roger Wattenhofer, Sumu Zhao, *Symbolic Music Genre Transfer with CycleGAN*, https://arxiv.org/abs/1809.07575.

[3] Chien-Yu Lu, Min-Xin Xue, Chia-Che Chang, Che-Rung Lee, Li Su, *Play as You Like: Timbre-enhanced Multi-modal Music Style Transfer*, https://arxiv.org/abs/1811.12214.