Music Generation using MIDI files

Benescu Matei-Alexandru

A neural networks project

Cuprins

[Abstract 3](#_Toc136293956)

[Introduction 4](#_Toc136293957)

[Methods 5](#_Toc136293958)

[Data and Preprocessing 7](#_Toc136293959)

[Results 9](#_Toc136293960)

# Abstract

Musical generation has always been considered a human trait, considering creativity uniquely human.

Considering the fact that music is often polyphonic, it adds to the challenge of generating music.

This project aims to explore automatic music generation using neural networks, specifically focusing on the application of recurrent neural networks (RNNs) and stacked long short-term memory (LSTM) networks with an attention mechanism. The project utilizes MIDI files from "The Legend of Zelda: Breath of the Wild" as the database for music generation. By employing the music21 and Keras libraries, the project aims to train a model that can learn from the MIDI data and generate new musical compositions.

# Introduction

Automatic music generation has gained significant interest in recent years due to advancements in neural network-based techniques. This project surveys the literature related to neural network-based music generation and draws inspiration from the book "Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play" by David Foster. The project aims to leverage the expressive capabilities of RNNs and stacked LSTM networks with an attention mechanism to create compelling and coherent music compositions.

# Methods

The chosen approach involves building an RNN-based model using stacked LSTM layers. The attention mechanism helps the model focus on relevant parts of the input sequence while generating music. The music21 library is used for data processing, enabling the conversion of MIDI files into a suitable numerical representation for training the neural network. The Keras library is employed for building and training the model, leveraging its extensive support for neural network architectures.

Recurrent Neural Networks (RNN)

A recurrent neural network is a class of artificial neural networks that make use of sequential information. They are called recurrent because they perform the same function for every single element of a sequence, with the result being dependent on previous computations. Whereas outputs are independent of previous computations in traditional neural networks.

LSTM layers are a variant of RNNs that incorporate a memory cell and gating mechanisms. The memory cell allows the network to store information over long sequences, enabling the capture of long-term dependencies. The key components of an LSTM layer are:

1. Cell State: The memory cell that carries information across time steps.
2. Input Gate: Controls the flow of information into the cell state.
3. Forget Gate: Determines what information to discard from the cell state.
4. Output Gate: Filters the information from the cell state to generate the output.
5. Hidden State: The output of the LSTM layer that carries information to the next time step.

The LSTM's gating mechanisms, such as the input, forget, and output gates, allow the network to selectively retain or discard information, addressing the vanishing gradient problem. This capability makes LSTMs particularly effective for modeling long-range dependencies in sequential data.

The code consists of two main files**: lstm.py** and **predict.py**, both contributing to the task of automatic music generation using LSTM-based neural networks.

In lstm.py, the **train\_network()** function prepares the MIDI file data and feeds it to the neural network for training. It first calls the **get\_notes()** function, which retrieves all the notes and chords from MIDI files in the specified directory. These notes are then processed and saved as a serialized file using the pickle module.

The **prepare\_sequences()** function in **lstm.py** takes the extracted notes and converts them into sequences that can be used by the neural network. It creates input sequences and corresponding outputs, where each input sequence represents a sequence of notes/chords, and the output is the next note/chord in the sequence.

The **create\_network()** function in lstm.py builds the structure of the neural network using the Keras library. It constructs a stacked LSTM model with multiple layers, including dropout and batch normalization for regularization. The model is compiled with the RMSprop optimizer and categorical cross-entropy loss.

Finally, the **train()** function in lstm.py trains the neural network using the prepared input sequences and outputs. It saves the best-performing weights during training using the ModelCheckpoint callback.

In **predict.py**, the **generate()** function generates new notes for a piano MIDI file using the trained neural network. It loads the serialized notes data and prepares the input sequences in a similar manner as in lstm.py. The **create\_network()** function is also used to build the network structure, and the trained weights are loaded.

The **generate\_notes()** function in predict.py generates a sequence of notes based on a starting pattern and the trained model. It iteratively predicts the next note, updates the pattern, and repeats the process to generate a series of notes.

Finally, the **create\_midi()** function converts the generated notes into note and chord objects and creates a MIDI file from these objects.

# Data and Preprocessing

The project utilizes a database of MIDI files extracted from "The Legend of Zelda: Breath of the Wild.". These files capture the musical essence of the game and provide a diverse range of melodies, harmonies, and rhythms. The MIDI files are preprocessed, converting them into a suitable format that can be fed into the neural network. Various musical features such as note sequences, chord progressions, and tempo information are extracted to facilitate the training process.

Music21

[Music21](http://web.mit.edu/music21/) is a Python toolkit used for computer-aided musicology. It allows us to teach the fundamentals of music theory, generate music examples and study music. The toolkit provides a simple interface to acquire the musical notation of MIDI files. Additionally, it allows us to create Note and Chord objects so that we can make our own MIDI files easily.

Keras

[Keras](https://keras.io/) is a high-level neural networks API that simplifies interactions with [Tensorflow](https://www.tensorflow.org/" \t "_blank). It was developed with a focus on enabling fast experimentation. We will use it for training the LSTM model.

# Results

* Using the following midi files:  
  Breath of the Wild - Great Fairy Fountain
* Breath of the Wild - Kass's Theme
* Breath of the Wild - Korok Forest (Day)
* Breath of the Wild - Main Theme
* Breath of the Wild - Mipha's Theme
* Breath of the Wild - Rito Village (Day)
* Breath of the Wild - Rito Village (Night)
* Breath of the Wild - Zora's Domain (Night)

I got the following results:

* The network was trained using a sequence length of 100, meaning it learned patterns from a window of 100 notes or chords to predict the next note or chord.
* Dropout layers with a rate of 0.3 were added to regularize the model and reduce overfitting.
* After training the network for 120 epochs, the generated notes were then converted into note and chord objects. For chords, the notes were split and assigned to the corresponding chord object. Finally, a MIDI file was created using the generated notes and chords.