Polyphonic music accompaniment for existing sheet music

Ruben Ganansia
Master of Engineering in Computer Science
Cornell University
New York, NY 10044
rgg64@cornell.edu

Raphael Rozenblum
Master of Engineering in Computer Science
Cornell University
New York, NY 10044
rr694@cornell.edu

Antoine Bertrand
Master of Engineering in ORIE
Cornell University
New York, NY 10044
ab2887@cornell.edu

Pierre Pasquet
Master of Engineering in ORIE
Cornell University
New York, NY 10044
pp492@cornell.edu

Samuel Salfati
Master of Engineering in ORIE
Cornell University
New York, NY 10044
sms723@cornell.edu

Abstract

Music accompaniment is one of the key problems liked to computer music generation. This project aims at offering a new model for piano accompaniment based on the MuseGAN model for multitrack polyphonic music generation. By identifying the benefits and flaws from this model, we hope to be able to create a model for music generation that will take into account this study. We made the code available at *https://github.com/RaphRozenblum/Musical-Accompaniment-GAN*.

Project Report for Cornell Tech Spring 2020 class of Deep Learning

1 Introduction

Creativity is a long-cherished and widely studied aspect of human behavior that allows us to "reinvent the familiar and to imagine the new" [1]. A recent area of creativity, named Computational Creativity, has emerged recently. The premises of this area are that computational modeling can yield important insights into "the fundamental capabilities of humans and machines" and claims that it is possible to construct autonomous systems that produce novel and useful outputs that are deserving of the label "creative" [1]. Creative Computation research can have a significant impact on many aspects of modern life, with real consequences for the worlds of art and entertainment. Alongside visual art and creative writing, musical composition is a core act of creativity that is considered uniquely human [2].

A growing area of application of deep learning is the generation of content. Content can be of various kinds: images, text and music, the latter being the focus of this project. In the context of computer-based music generation, the objective we are focusing on is the design and construction of autonomous music-making systems, as opposed to systems assisting human musicians (composers, arrangers, producers. etc).

The main motivation for using deep learning to generate musical content lies in its generality [3] which makes the system usable for various musical genres, as opposed to rule-based systems which have proven to be much more cumbersome to extend[4]. As a matter of fact, the research domain in deep learning-based music generation has welcomed big actors from the digital media and entertainment industries, such as Google (Magenta Research Project) and Spotify (Creator Technology Research Lab) as well as emerging startups such as Amper and Jukedeck that focus on the creation of original music for commercials and documentary.

The applications of a model that can create a multi-track accompaniment for an instrument are without a doubt manifold. Our goal here is to leverage the creative potential of MuseGAN and mix it with a conditionnal GAN (CGAN) to create an algorithm that can create accompaniments for a piano track using four instruments: guitar, strings, bass guitar and a drum set.

2 Background

In this section, we will elaborate on the various techniques, methods and models for music generation.

2.1 Deep Learning and Markov Models

Deep learning models are not the only models able to learn musical style from examples. Markov chain models are also widely used. In fact, their conceptual simplicity are widely appreciated and can easily be implemented with transition probability table. They also thrive with restricted amount of training data. However, first order Markov models do not capture long-term temporal structures and, while higher order models are possible, they require an exponential amount of data and can easily overfit the training dataset, leading to plagiarism : they do not generalize well.

On the other hand, while neural networks yield complex optimization algorithms, they generalize really well and can learn long-term and high-order dependencies through the use of distributed representations.

As deep learning methods are now mature, the advantages of using them overcome their limitations.

2.2 Recurrent Neural Networks and Generative Adversarial Networks

Recurrent Neural Networks thrive when they can operate on structure data. In the context of music, given a note, it can leverages the sequential nature of music notes to choose from a discrete set of possibilities for subsequent notes. In particular, the attention mechanism widely used for text data can be used in music: to understand what note or sequence of notes is likely to follow from a particular given passage, it is crucial to use information from far back in the sequence, i.e music score. In particular, a simple encoder-decoder can be implemented to model such a mechanism.

While the attention mechanism works well for single-line (monophonic) music, it is not suited for multiple lines of music. While it could generate each music line independently, we want to account for some sort of dependence between music lines (instruments), as in real life.

Generative Adversarial Networks models can accept multiple channels of music as individual streams and learn how they should interact with each other to generate sound and harmonious music. Such models attempt to model the process of musical organ.

2.3 Existing models

There are currently several existing music generation algorithm. OpenAI has started with MuseNet, a deep neural network that can generate 4-minute musical compositions with 10 different instruments, and can combine styles from country to Mozart to the Beatles. At its core, MuseNet is a recurrent neural network based on GPT-2, a large-scale transformer model trained to predict the next token in a sequence, whether audio or text. It uses the previously discussed attention mechanism [5]

More recently, OpenAI has released JukeDeck, a model based on VQ-VAE-2, an autoencoder model compresses audio to a discrete space, using a quantization-based approach. [6]

Finally, MuseGAN is a model based on Generative Adversarial Networks consisting of two parts of two parts: a multitrack model and a temporal model. The multitrack model is responsible for the multitrack interdependency, while temporal model handles the temporal dependency. The temporal model is two-fold: a part generating from scratch and an other accompanying a track given a priori by the user. [7]

3 Data

3.1 Basic music notions

A musical composition is composed of different tracks, usually one per instrument. The leading track is the one that dictates the melody and the overall style and groove of the composition. The other ones are accompaniment tracks.

A sheet music translates all sorts of musical notions into written instructions for each track, and can be thus very complex. To make its time structure more readable, it is divided into bars. A bar usually represents 4 units of time in a track, and can be divided virtually into time sub-intervals of equal length. Each note, or pitch, is placed between those bars, and between horizontal lines for the musician to know which one he has to play and when.

3.2 Tensor representation

We used the sheet music structure to translate a music composition into a tensor representation. For each track:

- $b_1, ..., b_l$ bars in the track
- $t_1, ..., t_m$ time sub-intervals in a bar
- $p_1, ..., p_n$ possible playable pitches per sub-interval

$\delta_{i,j,k} = 1$ if pitch p_k is played at time t_j in bar b_i, else $\delta_{i,j,k} = 0$.

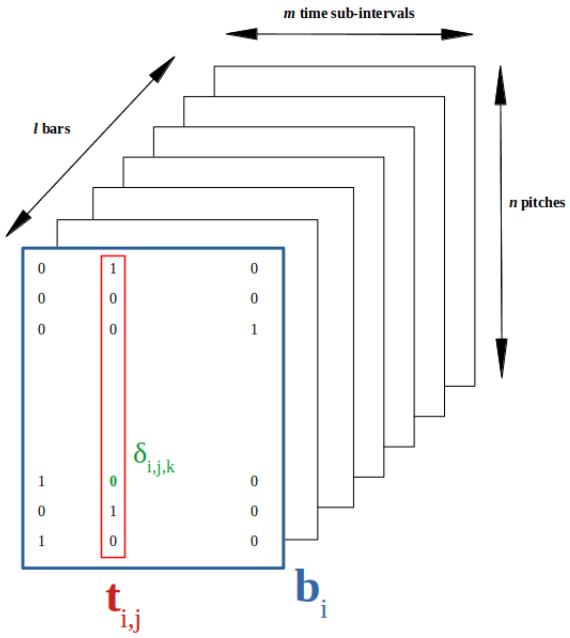

Figure 1: $l \times m \times n$ one hot tensor representation of **one** track.

Assuming it contains q tracks, a composition is a $q \times l \times m \times n$ tensor. For this project, we focused on piano conditional melodies. To simplify the problem, since there are 128 keys on a piano, we considered that there were only 128 possible pitches for any of the accompaniment tracks: $n = 128$. Each bar is divided into 96 sub-intervals and we limited the compositions to be 8-bar long: $m = 96$ and $l = 8$. Eventually, we decided to work on 5 tracks per composition (including the conditional melody), $q = 5$, with the convention that the first track is the melody.

3.3 Data set

We used the Lakh Pianoroll Dataset (LPD) [8] [9] which contains Multitrack Pianorolls. We decided to use songs only 21,425 songs as these songs contains one or less time signature change events, have a time signature of 4/4 and have their first beat starting from time zero.

Each song is stored in a MIDI file under a $k \times n \times q$ tensor, with $k =$ number of time steps in the song, $n = 128$ and $q = 5$. The MIDI files don't contain the bars dimension (l in the previous section representation), so we had to create it. As songs have a different number of time steps, we padded this dimension with 0s to make sure that the total number of time steps per song is divisible by 96, so that the number of bars is an integer. Dividing by 96 along k axis creates l bars. If l isn't divisible by 8 (as we used 8-bar long compositions), we padded the song with 0-bars. Eventually we broke down the l axis into 8-bar pseudo-compositions. Using this process on the 21,425 songs available, we were able to build around 200,000 parts of compositions with standardized dimensions $5 \times 8 \times 96 \times 128$. The melody of one composition is always the first track.

Also, these songs are matched with a high confidence score to songs from the Million Song Dataset (MSD). The MSD is a freely available collection of audio features and metadata for a million of contemporary popular music. This dataset is a cluster of sub-datasets containing a collection of features and metadata derived from the one million songs. Other components include genre labels, songs similarity, user-related data, etc.

Therefore, for each song in the LPD, we can define and store a wide range of attributes, from the simplest qualitative ones such as the name of the song, a reference to the parent album, the year it was released, as well as more quantitative user-defined attribute such as the "energy" level of the song, the "loudness" and the tempo.

4 Model

4.1 Global architecture

Our model is a complex GAN derived from MuseGAN. Like a classic GAN, it contains a generator module and a discriminator module. The generator has an intricate structure: it is made of several sub-modules to capture the temporality and the structural aspect of music.

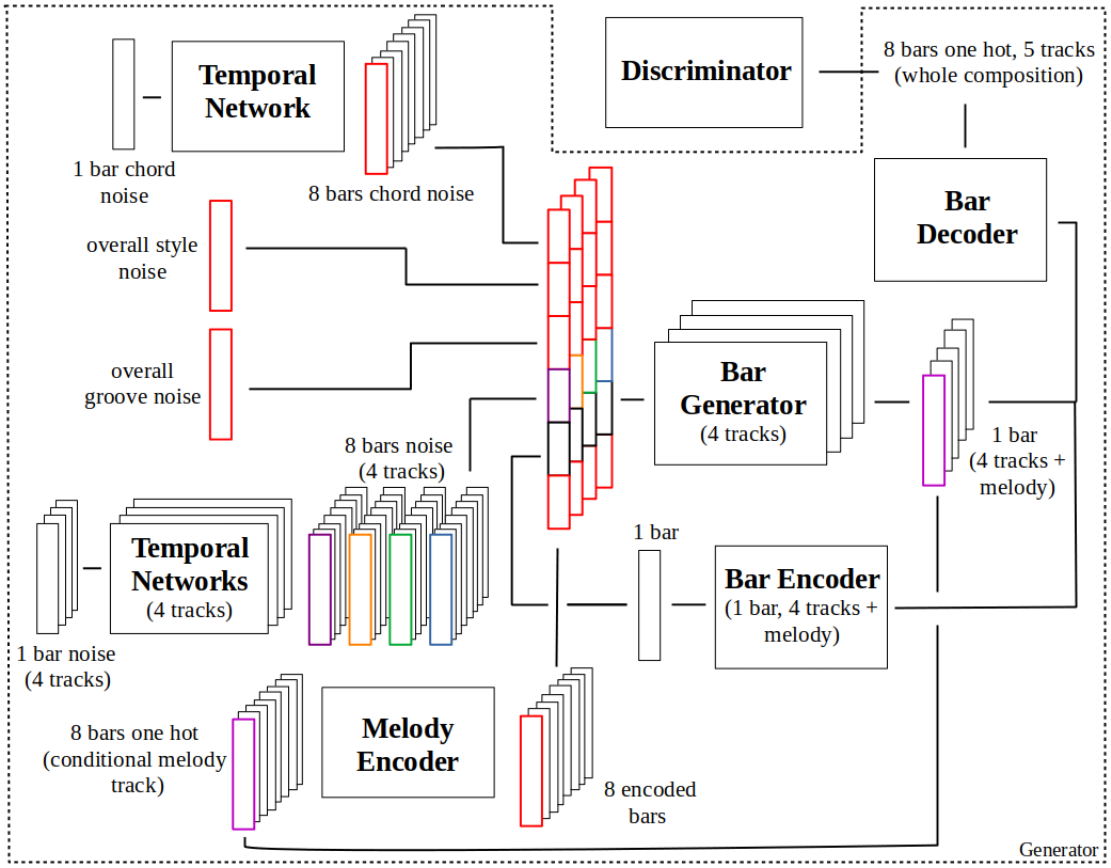

Figure 2: Global architecture.

4.2 Generator

4.2.1 Melody encoder

The goal of the melody encoder is to transform a $8 \times 96 \times 128$ one hot tensor representing the conditional track melody into a $1 \times 8 \times encoded\_melody\_dim$ float tensor, where $encoded\_melody\_dim$ is a hyper parameter. It is the conditional point of entry of the generator.

It is constituted of two 2D convolutional layers with ReLU activation, and one linear layer. The first convolution applies a kernel of (4, 12), 12 like the number of pitches in an octave. The second one applies a kernel of (3, 3). All weights are initialized using Kaiming algorithm, and both layers contain a Max Pooling of kernel (2, 2).

4.2.2 Temporal network

The temporal network module is used to extend temporally a bar into 8 bars. It takes a $1 \times z\_dim$ noise tensor as input and outputs a $1 \times 8 \times z\_dim$ noise tensor. Overall, 5 temporal network modules are used in the generator: we extend a 1-bar chord noise common to all tracks, and we extend a 1-bar accompaniment noise per track (4 accompaniment tracks so 4 modules). Each one of the modules learns separately from the others.

A temporal network is constituted of two convolutional layers with ReLU activation and batch norm, of kernels (2, 1) and (7, 1). All weights are initialized using Kaiming algorithm.

4.2.3 Bar generator

The bar generator is probably the heart of the generator. It generates the composition bar per bar, 4 tracks at a time. We use four different bar generator modules (one for each track). To generate a bar $\mathbf{b}_i$, each one of the modules will receive as input the concatenation of:

- the i^{th} bar generated by the chord temporal network,
- the i^{th} bar generated by the temporal network of the corresponding track,
- the i^{th} bar encoded by the melody encoder,
- a $1 \times z\_dim$ noise tensor representing the overall style of the composition,
- a $1 \times z\_dim$ noise tensor representing the overall groove of the composition where z_dim is a hyper parameter

To achieve coherence in the composition, we added a RNN unit: on top of the above list, we concatenated a $1 \times encoded\_bar\_dim$ tensor representing the previous generated bar, encoded by the bar encoder.

As a result, each one of the modules receives a $1 \times 4 * z\_dim + encoded\_melody\_dim + encoded\_bar\_dim$ tensor and outputs a $1 \times 96 \times 128$ tensor corresponding to a bar in a given track. To obtain the full bar with the 5 tracks, we concatenate the 4 generated bars and the original one hot track melody. The $5 \times 1 \times 96 \times 128$ full bar is then sent to the bar encoder to encode it and repeat the process for the next bar to generate. Between each step, the generated 5-track bar is also sent to the bar decoder.

A bar generator module is constituted by 10 layers. A first linear layer with a batch norm and ReLU activation, to cast the large input vector into 1024 dimensions. Then 9 transpose convolutional layers with batch norm and ReLU activation. The first 5 transpose convolutional layers divide the size by two along the step-per-bar axis each time, with (2, 1) kernels, and the sixth one with (3, 1) ensures a 96-dimension along this axis. The last three layers extend the bar in the pitches axis, with respectively (1, 4), (1, 4) and (1, 12) kernels to achieve a dimension of 128 along this axis. Again, we use 12 to reproduce the inverse of the melody encoder, and 12 like the number of pitches in an octave. All weights are initialized using Kaiming algorithm.

4.2.4 Bar encoder

The bar encoder is used to encode a $5 \times 1 \times 96 \times 128$ tensor representing a 5-track bar into a $1 \times encoded\_bar\_dim$ representation, where $encoded\_bar\_dim$ is a hyper parameter. Its output is used to generate the next bar, by concatenating it with the input vectors of the bar generator modules, as mentioned in the previous section.

The bar encoder is constituted of two convolutional layers with leaky ReLU activation, of kernels (3, 12) and (2, 3). All weights are initialized using Kaiming algorithm, and both layers contain a Max Pooling of kernel (2, 1). The last layer is a linear layer to cast the output into the desired dimension.

4.2.5 Bar decoder

The bar decoder is the final module of the generator. While the bar generator generates the composition bar per bar, it aggregates those 5-track bars to compose the 8-bar composition, and transform them into one hot tensors. The process is very simple: the bars of each tracks are normalized, and, given a track, $\delta_{i,j,k}$ is set to 1 if $b_{i,j,k}$ is above a threshold θ, where θ is a hyper parameter. It outputs a whole composition represented by a $5 \times 8 \times 96 \times 128$ one hot tensor. The latter is sent to the discriminator.

4.3 Discriminator

The discriminator - or Critic since this is a **WGAN** - is, as is often the case, less complex than the Generator. The first part is a convolutional base using 3D convolutional layers and then a simple fully connected classifier to a single node **with no activation at the end**.

In the convolutional base, kernels are kept small except for the first one in the pitch dimension where a kernel of size 12 is used. This is because 12 notes are equal to an octave, which is an important notion in music theory.

5 Evaluation and Results

5.1 Training

The model was trained using the **WGAN algorithm** as well as a **gradient penalty loss** :

$$L_{GP} = \left[1 - \left|\left|\nabla D(\frac{\alpha * RealSong + (1-\alpha) * GeneratedSong}{2})\right|\right|_2\right]^2$$

Where $\alpha \sim U(0, 1)$ to create a random average.

Weights of the Discriminator were clipped in the range [-0.01, 0.01].

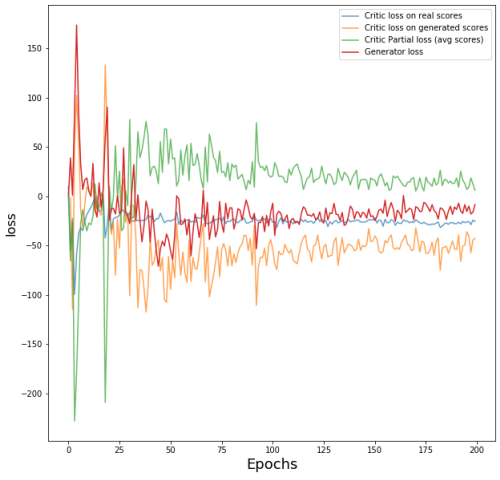

Figure 3: Losses

Convergence was especially challenging but using 5 critic loops per batch proved to solve the issue. We trained using Google Colab Pro with CUDA.

5.2 Results

Accompaniment for piano gave this result:

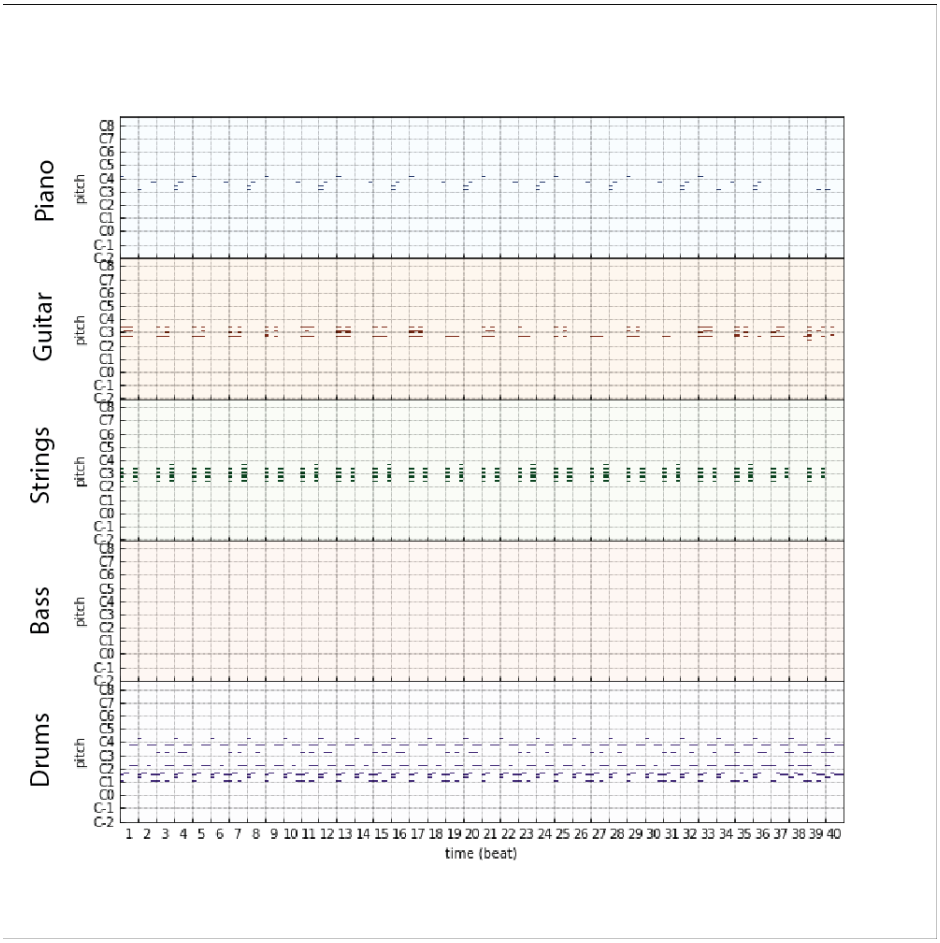

Figure 4: Pianorolls of the results, the first pianoroll is the conditional track

This result used a threshold value of 0.4 on the output of the tanh activation of the Bar Generators.

5.3 Analysis

The model's output sounds globally harmonious. A great achievement of this model is that it learned to play all the instruments in the right chord. Guitar and Strings are in the right range of pitches and have clear rythmic patterns.

For guitar and strings, the model learned how to create simple chords with three to four notes, which is a widely used way to play chords.

There is some mode collapse in our creation. It seems that most of the time the model will use the same major chords and almost always the model will completely skip the bass track. The drum pattern is also always very similar.

References

[1] F. A. C. Tony Veale, "Computational creativity, the philosophy and engineering of autonomously creative systems," 2019.

[2] D. Foster, "Generative deep learning : Teaching machines to paint, write, compose and play," 2019.

[3] F.-D. P. Jean-Pierre Briot, Gaetan Hadjeres, "Deep learning techniques for music generation," 2020.

[4] K. Ebcioglu, "An expert system for harmonizing four-part chorales," 1988.

[5] Musenet, "https://openai.com/blog/musenet/](https://openai.com/blog/musenet."

[6] Jukedeck, "https://openai.com/blog/jukebox/](https://openai.com/blog/jukebox/)."

[7] Musegan, "https://salu133445.github.io/musegan/model](https://salu133445.github.io/musegan/model)."

[8] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.