Deep Musician

Automatic Music Generation using Deep Learning

Fabian Gabelberger

January 18, 2023

Abstract

The main idea of this project is to create a model that is capable of automatically generating new and unheard musical melodies that resemble human ones. This is achieved with a deep neural network that was trained with a large amount existing MIDI files. The network uses a sequence aware Encoder-Decoder structure that is capable of creating sequences of notes of arbitrary length. The encoder and decoder each consist of a 2-layer GRU network, whereas the decoder has an additional classifier.

1 Introduction

With *MuseNet OpenAI* created 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." [1] Behind this project resides the (philosophical) idea that musical compositions can arise not only from a particular (abstract) artistic understanding of harmony, rhythm, melody, etc., but also *solely* from a variety of previous works that are incorporated into the new, unheard piece as a wealth of experience.

MuseNet is fuelled by ""a large-scale *transformer model* trained to predict the next note(s) in a sequence." [1] While this approach using a transformer model is certainly state of the art for sequential data and produces a truly vibrant and rich musical style, it is very resource intensive and can thus – simply due to hardware limitations – not be the scope of a project for *Applied* Deep Learning .

In contrast, there are also more lightweight approaches that use a variety of RNN structures (such as LSTM models or RNN [2–6] with Self-Attention [7] and even a CNN [8] - which is quite interesting considering the temporal dimension of music that CNN are not designed to depict. (For a general overview of the different approaches of generative music models see: [9].)

2 The Problem

2.1 A philosophical Problem?

All these ideas try to solve the problem of how to outsource the creation of music – which until now has always been in the hands of man – to the machine. In terms of music history, this is not necessarily about creating original works that represent a new avant-garde in music. From a purely pragmatic point of view, such a model could rather offer musicians the possibility to complement their works (be it with certain instruments or as a source of ideas) or to generally grant artists access to freely available and and non-proprietary music in order to easily create background music for a video or other content, for example.

At the bottom of this question itself, lies the problem of the creative and generative abilities of machines, algorithms, models etc. Since machines are said to produce merely repetitive patterns, it is hard to imagine how true creative output is to emerge from them. Yet the recent successes of deep (reinforcement) neural models[1] has shown, that the boundary between human and machine creation and creativity is getting ever thinner and thinner.[2]

[1]The creative/machine generation field has gained tremendous traction through its successes in recent years, from Deepmind's AlphaZero, which plays chess, Go, and Shogi, to various image processing and generation software, to OpenAI's latest ChatBot. The list is constantly being refreshed and expanded, and it seems we've only just exhausted the first stages of machines' creative potential.

[2]Especially if we acknowledge that the boundary between repetition and creativity is not a strict criterion, but a fluid line, where both sides are mutually dependent.

While I neither address nor answer these philosophical questions on a theoretical level in the project, I will make a first delicate attempt to explore machine-creative ways by means of a practical application. So the point is not to find a theoretical answer to the question of creativity, but to show by means of a practical application to what extent machine learning produces musical pieces that we as humans grasp as melodious, beautiful, and and ultimately human. Thus reformulating the former existential question to a rather categorical and technical one: How do the creations of machines differ from that of their human ”counterparts” - and are we able to tell the difference?

2.2 The concrete problem

Concretely, i.e. looking at the practical application and thus at the field of computer science and artificial intelligence, the problem that such an algorithm needs to tackle is that it not only needs to make a prediction about the further course of a musical piece, but to design such a piece itself. While a large part of neural networks is trained to classify different things or to deliver the corresponding output for a given input (be it a classification, a translation or similar), this application is about *creating* something (”without” a corresponding input). In addition, musical pieces are sequences that have a temporal component at their core, i.e. their elements build on each other in time and do not exist synchronously or simultaneously in the moment.

Two important elements that we therefore have to take into account when dealing with the topic of machine-generated music are:

1. Sequence / temporal awareness
2. Generation instead of classification

3 The Solution

The first point is solved by certain neural architectures (in particular RNN, LSTM and Attention Mechanism) that allow to integrate a temporal structure into the machine learning process. As a result, the individual training data are no longer (temporally) independent data points, but become a coherent sequence whose course has to be traced, i.e. learned, by the algorithm. The algorithm is given the steps in chronological order and its task is to correctly predict the next step in time. During training, the model learns to correctly continue the given sequences.

The second step is approached by not continuing existing melodies correctly after training, but by animating the model to predict new (own) melodies. This is done by passing an empty input to the model on the basis of which the network predicts the temporally following second element and on the basis of these two elements the third and so on.

In the introduction we have heard about different and very potent architectures that are sequence aware and capable of generation. Although I would have loved to implement an Attention Mechanism, I decided to start on a much smaller scale with a rather simple encoder-decoder architecture, that each consist auf a two-layered GRU structure. My project is thus dedicated to a small section of the topic of creative algorithms, namely the automatic generation of music, by implementing a sequence aware model, that is capable of generating musical sequences of arbitrary length.[3]

Generative algorithms that attempt to perform an artistic feat are almost exclusively implemented as neural networks. This is simply because they are the strongest and most comprehensive algorithms and can produce new data that did not occur during their training. Of course, it would also be possible to generate melodies via rule-based algorithms. However, these have the disadvantage of always following a certain rule or pattern and do not manage to surprise the user. The melody will always run within a certain framework, but will never have a ”creative” potential beyond what has already been seen. Moreover, previous experiments in the literature have shown that the approach of creating generative melodies using neural networks is very successful. Of course, it would also be exciting to implement this reinforcement learning and to train the algorithm via positive and negative incentives. However, the problem here is that we would need a measurable criterion for beautiful and harmonic melodies. But to be frank, due to hardware limitations, the melodies my model produced so far are similar to simple based melodies, which are not very interesting. Yet, this should be solved with more powerful hardware.

[3]I realize that I am only scratching the surface of the issue given the incredibly vast and comprehensive models.

4 Outline of the Project

4.1 Goal

The goal of the project is to create a model that produces melodies that sound human, or natural, i.e., not mechanical. Making this goal mathematically quantifiable is not trivial, since there are a variety of ways to represent notes and sequences of notes that each require different metrics. In addition most of these metrics are not able to capture the essence of a creative sequence of notes. Of course, there are metrics that describe how well an algorithm predicts a certain sequence, but as we will see below, these have weighty drawbacks. Therefore, I remain with the approach of measuring generated melodies according to simply my human ear.

4.2 Data Structure

4.2.1 Representing Music

The basis of music is formed by sequentially played sounds or tones that can be represented as a complex *waveform*. These individual sounds can be joined together in any way to form an entire piece of music, which in turn is again a single waveform that we can play back and listen to in different audio formats (MP3, FLAC, WAV etc.).

4.2.2 MIDI

While this form of representation already depicts a concrete shaping of the music in the form of an unique audio file, it is also possible to specify the individual tones of the piece in the form of notes with different parameters. The advantage here is that the concrete instrumentation is abstracted from and only the internal *structure* of the piece is considered. The generally accepted standard for this representation is *MIDI*. By means of MIDI it is possible to transmit not only the pitch and length of the individual notes, but also other parameters such as velocity - yet, no concrete waveform is produced.

Thus, due to its abstract nature MIDI offers the possibility to extend the input of the model successively. While initially only monophonic audio tracks with constant dynamics and tone length are used, these parameters are to be successively added to the input to see how the created melodies change.

4.2.3 Preprocessing

However, neural networks cannot be trained with midi files themselves. Therefore, these must be converted into another form in which the notes can be passed to the network. There are a lot of possibilities for this, of which I have chosen a rather classical and simple one: The piano roll. Here a 2 dimensional matrix is spanned, whose x-axis represents the time and whose y-axis represents the 88 notes of the piano. Each touch of a note is marked with 1 in the matrix at the corresponding time t - all other cells remain empty (0). This representation is very clear and intuitive. However, it has a big problem: since only a small percentage of the available cells are filled – i.e. most of the time NO note is played – there is a big imbalance in the data. Here other representations such as the language based ones would have definitely been better.

4.3 Metric and Loss

Although the final goal of the model is to generate melodies that sound human, we need to define a metric and loss that quantifies this. This is not trivial, since classical metrics have difficulty dealing with this problem and return suggestive and misleading values and classical losses do not optimise for the desired goal, that is the generation of a human sounding melody.

I faced this problem during the later stages of my experiment, when the model easily learned according to the classical BCE-loss, but afterwards during testing only generated empty melodies. This was due to the fact, that it actually guessed almost all of the notes correctly as they were not being played. So the empty sequence resembled the input it was given most of the time. Or put, differently the model was stuck in a local minimum.

4.3.1 Focal loss

I learned that image classification faces a similar problem and solves this by using a so called focal loss. Focal loss is a loss function that is used in image classification tasks, particularly those involving object detection. The main idea behind focal loss is to down-weight the contribution of easy examples in the training data and focus more on the hard examples, which are typically the ones that are more challenging to classify correctly. This is achieved by modifying the standard cross-entropy loss function by introducing a weighting term that increases the loss for easy examples and decreases the loss for hard examples. The result is a loss function that is more ”focal” on the hard examples and helps the model to better learn from them and improve its performance on the task.

With the introduction of focal loss in my model it started generating meaningful sequences of notes. To keep track of the validity of the generation of sequences I also introduced a density metric, that measures the average notes played per time step. What is more, lowering the learning rate also helped a lot in not getting stuck in a local minimum.

Yet, the two parameters of the focal loss (alpha and gamma) need to be carefully adjusted to obtain meaningful results, which for this project has not been accomplished so far.

4.4 Architecture

The architecture of the network consists of an encoder-decoder structure and a subsequent classifier. The input (a piano roll with length 96) is first passed through the encoder – only the returned hidden state is of importance here, the output itself is discarded. Then an empty input (SOS-token) with the hidden state is passed to the decoder. This gives a new hidden state and a prediction about the next state. This prediction is then passed through the classifier and serves as a new input for the decoder together with the updated hidden state. This process runs over the entire length of the sequence, meaning that we iterate through the sequence length by starting with the SOS-Token, then using the previous output and hidden state as new input and. So while the encoder processes the incoming sequence all at once, the decoder and classifier build up the new sequence step by step.

- EncoderRNN: GRU(input: 88, hidden: 512, layers=2, dropout=0.2)
- DecoderRNN: GRU(input: 88, hidden: 512, layers=2, dropout=0.2)
- Classifier
  - Linear(in_features=512, out_features=256, bias=True)
  - ReLU
  - Dropout(p=0.5)
  - Linear(in_features=256, out_features=88, bias=True)
  - Sigmoid

5 Main Takeaways

The three most important takeaways of the project for me concern preprocessing, the evaluation and hardware.

5.1 Preprocessing

Here I learned in particular that the preprocessing part can exceed the other parts of the project by a wide margin in terms of time. So far I have worked in the field of machine learning especially with text, audiovisual data and numerical values. However, the symbolic representation of music was previously unknown to me from a computer science perspective. While I know that preprocessing from the other domains can also be time consuming, I was just overwhelmed by the wealth of possibilities with symbolic music and the time I needed to invest in order to obtain a meaningful representation of the music. Here it was important to keep a good overview and not get lost in the possibilities. The individual advantages and disadvantages may not be evident at first, but if you start with a simple approach, you can slowly work your way up and document the differences and improvements. It is important to keep in mind, however, that the different forms of representation are often so different that they cannot simply be fed into the existing network - for example, if one chooses a text-based representation as opposed to a piano roll.

Linked to this is also the insight that there can be an unbelievable number of approaches to solving a problem, which already begin during preprocessing and not just during modeling. One often has the feeling that the differences only become apparent or noticeable during modeling. However, I learned that it can be important to always take a step back and look at the problem with new eyes and possibly even change the underlying data structure to achieve your goal. It was really impressive to see the sheer number of ways that symbolic music can be represented (from classical piano rolls to linguistic models), all of which highlight different aspects and provide benefits. I would have liked to try out more of these approaches - but in the end I was glad to have limited myself to only one.

5.2 Evaluation

The most important lesson in terms of evaluation and training I learned about metrics and loss is that a model that is almost perfect (in terms of certain metrics) may still produce meaningless results. A high Accuracy or a F1 value may still mean that the desired results are not achieved. This is due to the fact that these metrics only represent an abstract value and abstract from the concrete situation. Therefore, it is always important - even if sometimes tedious - to look at concrete examples in the evaluation and not just to stay with the abstract metrics, as these can often be deceptive. In my case, as described above, my model had a very high accuracy, because the generated melodies were simply empty and therefore almost all notes were predicted correctly. The fact that this output completely contradicted the original approach is of course irrelevant for the model - it does what it is told to do. By changing the loss function, I was able to counteract this imbalance in the data (to a certain extent). In any case, the insight remains that in future projects I will pay even closer attention to the metrics used and the loss, and that I will always check the results with concrete individual cases during the evaluation and not rely merely on the simple metrics of the model.

5.3 Hardware

In addition to the aforementioned problem of empty melodies, I also had to struggle in particular with the limitations of my hardware, which does NOT include an Nvidia graphics card. To train an epoch merely with Mozart's sonatas with my CPU takes me a little more than half an hour. However, in this field of research, training over several hundred epochs is not uncommon. Because of this limitation, I have only ever been able to carry out my experiments with even smaller samples and a few epochs, which of course greatly distorts the results. This problem can either be solved by reverting to a simpler model (which is sometimes not possible) or to outsource the computation to a different machine, e.g. cloud computing service.

6 Workload

Task	estimated	actual
Dataset collection	7	12
Exploring, analysing and preparing data	12	45
Designing and building an appropriate network	25	40
Training and fine-tuning that network	15	17
Building an application to present the results	20	21
Writing the final report	8	6
Preparing the presentation of your work	5	5
Sum	92	147

Table 1: Workload

Overall, I invested an above-average amount of time in the project. On the one hand, this is due to the fact that I had an incredible amount of fun and enjoyment with the project, since it encompasses the entire process from the idea to the deployment and you can shape and follow the entire development. Also, the fact that we were encouraged to structure the project cleanly and neatly, to include automated tests and to generally orientate ourselves to the current business workflow and not just deliver a Jupyter notebook, I found very inspiring and justified for me to invest more time than planned.

On the other hand, and this is also reflected in the divergence between the estimated and actual timeline, I especially underestimated the data preparation and preprocessing. This is certainly due to the underlying data structure (MIDI), which cannot be easily processed or used for machine learning - many preprocessing steps were required here. In addition, I was initially overwhelmed by the sheer amount of options for representing MIDI data and researched for a long time before deciding on one. In retrospect, it would have made more sense to choose a simple approach from the beginning and implement it as soon as possible to have a working pipeline. In general, this is one of the main lessons I take away from the project: implement a simple pipeline (baseline) quickly at the beginning and develop further based on it, instead of starting with the biggest step and implementing everything perfectly (state of the art) from the beginning.

The second major divergence between estimated and actual timeline is found in designing and building an appropriate network. This is especially due to the fact that I have never implemented a generative algorithm using RNNs before and wanted to get to know the structure in more detail to create a kind of blueprint for myself that I can use for further projects. In addition, I wanted to work with Pytorch Lightning, which I have also never used before, as it eliminates the need for a lot of boilerplate code. So the time difference in the network designing process is especially also due to learning new framworks and techniques as well as building a well documented blueprint and could have been reduced for sure. However, it was worth it to me as it will save me additional time in the future.

So in summary, the next time I would start with a reduced approach and work my way up from there in small steps instead of doing everything in one go. This not only has the advantage that you quickly have a baseline (and a corresponding sense of achievement), but also that you can track the individual steps and improvements more precisely and thus have a better comparison between the models. Otherwise, I am very satisfied with my way of working and structure and would also try to organize it accordingly well for further projects.

References

[1] Christine Payne. *MuseNet*. URL: `openai.com/blog/musenet`.

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription*. 2012. DOI: `10.48550/ARXIV.1206.6392`. URL: `https://arxiv.org/abs/1206.6392`.

[3] Maria Klara Jędrzejewska, Adrian Zjawiński, and Bartlomiej Stasiak. "Generating Musical Expression of MIDI Music with LSTM Neural Network". In: *2018 11th International Conference on Human System Interaction (HSI)*. 2018, pp. 132–138. DOI: `10.1109/HSI.2018.8431033`.

[4] Nabil Hewahi, Salman AlSaigal, and Sulaiman AlJanahi. "Generation of music pieces using machine learning: long short-term memory neural networks approach". In: *Arab Journal of Basic and Applied Sciences* 26.1 (2019), pp. 397–413. DOI: `10.1080/25765299.2019.1649972`. URL: `https://doi.org/10.1080/25765299.2019.1649972`.

[5] Adrien Ycart and Emmanouil Benetos. "A Study on LSTM Networks for Polyphonic Music Sequence Modelling". In: *International Society for Music Information Retrieval Conference*. 2017.

[6] Sanidhya Mangal, Rahul Modak, and Poorva Joshi. "LSTM Based Music Generation System". In: *IARJSET* 6.5 (May 2019), pp. 47–54. DOI: `10.17148/iarjset.2019.6508`. URL: `https://doi.org/10.17148%2Fiarjset.2019.6508`.

[7] Akash Jagannathan et al. "Original Music Generation using Recurrent Neural Networks with Self-Attention". In: *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*. 2022, pp. 56–63. DOI: `10.1109/AITest55621.2022.00017`.

[8] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. *MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation*. 2017. DOI: `10.48550/ARXIV.1703.10847`. URL: `https://arxiv.org/abs/1703.10847`.

[9] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – A Survey*. 2017. DOI: `10.48550/ARXIV.1709.01620`. URL: `https://arxiv.org/abs/1709.01620`.