Deep Learning Techniques for Music Generation

01-19-2021

Elias Aouad
elias.aouad@student.ecp.fr

Yassine Abbahaddou
yassine.abbahaddou@student.ecp.fr

Abstract

Generating harmonic music is a complex task. Not only is it hard to build a model that generates smooth and harmonic music, but it is also hard to evaluate the generated music, since only humans are able to distinguish between real and fake music. In this paper, we will mostly use ABC encoded music as dataset. Our goal will be to be able to try models that generate music, and compare them with real music. We implemented several pipelines to generate new music. To evaluate the generated music, we will make use of BLEU score on ABC encoded music, and the discriminator's error rate on the GAN model. We also added to those metrics the human evaluation, as we asked for 20 volunteers to rate generated samples from each model.

1. Introduction

Mobile phone development and innovation driven by Tech companies have strongly impacted people's access to music and the relationship they entertain with it. From initially the need to be at home with a cassette or at a concert, the digitalization of access to music through apps and streaming platforms creates new use cases for music, increasing therefore the weekly number of hours spent listening to music. Indeed, the average US citizen spends on average 26.9 hours per week listening to music in 2019 according to Statista. This shows the importance of music to people and the strong connection they have with music sources.

A question related to how a human brain interprets music segments arises. What makes music more enjoyable than a random sound? Music is in fact like every other sound from a physics perspective: a signal composed of certain frequencies our ears could perceive and send to the brain for interpretation and decoding of the logical chronological sequence it receives.

However, what differentiates music from any other random noise is the set of notes that are being played, as well as the harmony in the played music that is gentler to the ear. It's not just about the notes that are being played, or the instruments that are being used, but rather a combination of all these parameters that come together to create one unique symphony. Cords, tone, frequencies, those are all parameters that enter in the complex process of generating a music.

Music is therefore a complex phenomenon. Understanding its components and patterns fascinates scientists, as it could open the door to some magnificent applications. We could for example mention music therapy, a new therapy technique used to cure patients with some underlying sickness condition, which could be either physical or psychological, but also some more lucrative applications that would be possible thanks to the emergence of artificial intelligence and the collection of large datasets.

Indeed, artificial intelligence aims at understanding and replicating the brain's functioning, and this could apply to music analysis. Using AI and more precisely algorithms based on neural networks to understand underlying patterns and features in music segments, so they could later generate their own technology-based music segments. A groundbreaking application we could see would be for Spotify or Deezer to start generating new music segments based on user preferences by replicating some underlying patterns. The large breadth of applications has therefore increasingly attracted scientists to explore this field.

Multiple studies have been conducted on generating random samples of music by training a model on a sufficiently large number of music samples, the objective being to generate new and original music. In this paper, we are going to study some of those models, and see what kind of music they output.

We have identified two ways to train these models. On way is to take samples of music as sound records and gener-

ate new music directly as a sound. On the other hand, other pipelines encode music using the ABC notation, which basically transforms music into text. Therefore, music is treated as a text, hence textual language processing methods can be applied on such type of dataset. In our study, we decided to go with this type of dataset; *ie* we had our music already encoded into text, except for training the GAN which required raw audio data.

2. Problem Definition

In this paper, we will focus on using deep learning techniques for generating music which would sound as natural as possible. To do so, we will make use of a dataset of piano music already encoded in ABC notation, which is one of the possible ways to write down music. Since the music is encoded into textual data, our objective will be to make use of textual processing methods, commonly used in natural language processing, to generate new music out of a given random seed. The difficulty of the problem arises in the evaluation part, since only human are able to discriminate between real and generated music. However, we will make use of BLEU score to evaluate the similarity of a generated music to real music.

3. Related Work

There has been a lot of work based on Recurrent Neural Networks for generating music. On of them in particular, is made by Allen Huang and Raymond Wu [1]. They made use of two different types of representation for music files, namely midi files and piano roll files, which were separated into two different datasets in order to train the model seperately on each music representation. They extracted their entire dataset from the MuseData website, and used a 2-layered Long Short Term Memory (LSTM) recurrent neural network (RNN) architecture to produce a character level model to predict the next note in a sequence.

In their methodology, they used two different tokenization techniques for each one of the representation datasets, and each token was assigned to an embedding, which would be tuned at the same time as the model. Hence, for a given sequence, what they would do is start by tokenizing the sequence, then compute embeddings for each token, and then concatenate the tokens into a list of embeddings which would result in the time sequence input that will be fed into the 2-layer LSTM.

Then, the output of the LSTM passes trough a softmax layer in order to produce probabilities, which will allow the model to choose the next note in the sequence. The loss is computed using cross-entropy loss, and they clipped their gradients during training to prevent them from exploding.

Hence, to generate a musical sequence, they would first choose a random seed, then generate the following token. Once they have generated this token, they add it to the seed, and then feed the network the new seed from which they remove the first token to keep the same size for the seed at the entry of the network. They do it repeatedly until the end of the loop. Once the loop ends, they output the generated seed.

In the end, to evaluate their work, they asked for volunteers to rate three generated samples of music between 0 and 10. The average rating suggests that the generated music is better than average, which suggests that they did a good job, considering that the music is plausible and can pass for human music.

Another approach was proposed by Rachel Manzelli et al. in their paper [4]. In their work, they tried to make a balance between raw audio models and symbolic models. Raw audio models are the ones that treat music as an audio, while symbolic models are the ones that treat music as a sequence of notes.

Their idea was to consider a Long Short Term Memory network to generate a symbolic structure of music, and then use it as a conditioning input to a WaveNet-based raw audio generator. Recall that WaveNet is a model developed by DeepMind to generate music from raw audio music [6]. In other words, they used the symbolic generation as a second time series input that would influence the music generation by the WaveNet model. As for evaluation, they observed that using this pipeline lowers slightly the training loss function in comparison with the original WaveNet model, and that is due to the additional symbolic embeddings that are used as input. However, they did not show qualitative evaluation as Allen Huang and Raymond Wu did in their paper, though their generated music is available on their website for subjective evaluation.

4. Data

4.1. ABC notation

ABC notation is a shorthand text-based form of musical notation that uses the letters A through G to represent musical notes, and other elements to place added values.

It indicate various aspects of the tune such as :

- **X :** denotes a reference number
- **M :** denotes Meter
- **K :** denotes the Key

Figure 1. Example of ABC notes

```
X: 1
T: Cooley's
M: 4/4
L: 1/8
R: reel
K: Emin
|:D2|EB{c}BA B2 EB|~B2 AB dBAG|FDAD BDAD|FDAD dAFD|
EBBA B2 EB|B2 AB defg|afe^c dBAF|DEFD E2:|
|:gf|eB B2 efge|eB B2 gedB|A2 FA DAFA|A2 FA defg|
eB B2 eBgB|eB B2 defg|afe^c dBAF|DEFD E2 :|
```

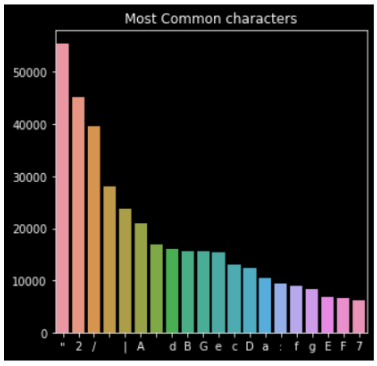
- **L :** denotes the beats
- **T :** denotes the Title
- **Z :** denotes the transcription The rest are notes that denote the melody for the song.

4.2. Data processing

We used the **Nottingham Music Database** [5] maintained by Eric Foxley that contains over 1000 Folk Tunes. Each tune consists of a header and a body. The header, which is composed of information fields, should start with an X (reference number) field followed by a T (title) field and finish with a K (key) field. The body of the tune in abc notation should follow immediately after. Tunes are separated by blank lines.

The data is currently stored in a character-based categorical format. We transform the data into an integer-based numerical format : each character is mapped to a unique integer.

Figure 2. Frequency of the most common characters



5. Experiments

5.1. Using GRU model

As already mentioned, the music is encoded with ABC notation. Hence, it can be treated as text. The first method that we applied was to make use of GRU to generate a sequence following a seed sequence that we feed to our model. This method is actually very similar to a translation task from one language to another, but in that scenario, we are trying to predict the next sequence of characters from the first sequence of characters.

First, we consider a 10-character sequence as input of the model, and we tokenize it into characters. Each character is assigned to an embedding of size 256, which will be tuned with the model. Then, each embedding passes through a GRU, which, at the end of the sequence, will output a hidden representation of the entire input sequence. The hope is that this representation of fixed size 256 will effectively summarize the entire sequence.

That first part is referred to as the encoder part. Next step, decoding the new sequence. The decoder is also based on GRU, it works as follows. It takes the last character of the sequence, embed it, and pass it through a ReLU activation. It also takes the previous hidden state of the GRU, and, they both pass through the GRU. The GRU will then output one vector of the same size as our vocabulary, which we softmax to get probabilities and output the character with highest probability. It will also output the next hidden state of our sequence. It loops these steps until maximum length of sequence is achieved. In our case, we generate a 10 character sequence and stop.

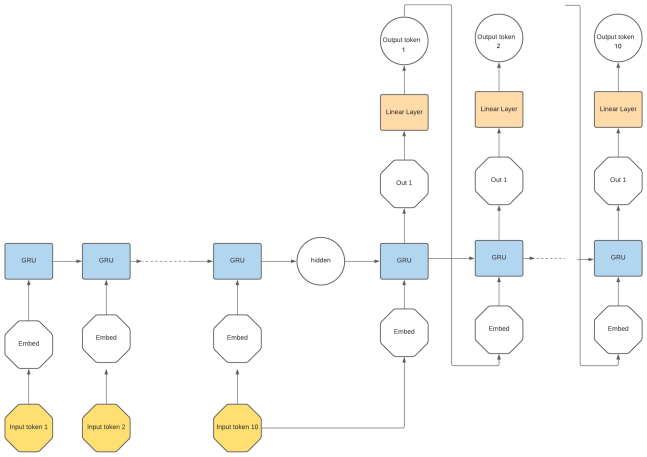
However, 10 characters are not sufficient in order to enjoy a good portion of music. Even though the 10 character limit was specified to ensure a better training, it does not produce a sufficient amount of music. Nonetheless, what we can do is create a loop in which each generated sequence is then used as a new input of the model to generate the next 10-character sequence. Hence, we loop until a sufficient amount of music is generated.

5.2. Using 2-layer LSTM model

Second method applied was to make use of LSTMs to generate every character of the song, an approach similar to [1]. First, the processing of the data is the same as the one used in the last method. However, the encoding/decoding process is not the same here, as we are going to make use of a 2-layer LSTM to process our sequence. The goal will be to output one hidden representation of size 256 for our sequence, then pass it through a linear layer, which after softmax will output probabilities for each output token. We would then have to choose the token with the highest probability.

To do so, we first pass our sequence through a first layer of LSTM. Each input token will pass through a LSTM cell,

Figure 3. Simplified scheme of the GRU model

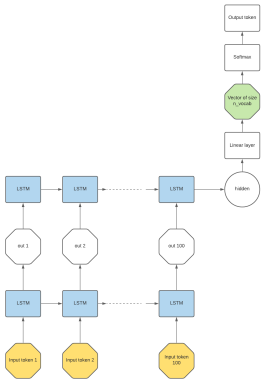


which will combine this input with the precedent hidden states of the sequence, and will output a vector of size 256. We will call that vector the encoder output.

We have as many encoder outputs as input tokens, hence we can treat this output as a new sequence. Each encoder output will pass through a new layer of LSTMs, and this time, the only output that interests us is the hidden representation of the sequence at the last LSTM cell.

Hence, we take this hidden representation of size 256 and pass it through a linear layer which will map it to a vector of size the length of our vocabulary. When we softmax this vector, we will get probabilities, and the output token is the token with he highest probability.

Figure 4. Simplified scheme of the 2-layer LSTM model



5.3. Using a many-to-many RNN with the attention mechanism

Our input and output are sequences of characters, respectively $x = (x_1, ..., x_{T_x})$ and $y = (y_1, ..., y_{T_y})$. x and y are usually referred to as the source and target notes. Our encoder is a non-stacked unidirectional RNN with GRU units. Our decoder is a non-stacked unidirectional RNN. It is a neural language model conditioned not only on the previously generated target characters but also on the source notes. More precisely, it generates the target notes which are the rest of the source notes $y = (y_1, ..., y_{T_y})$ one character y_t at a time based on the distribution:

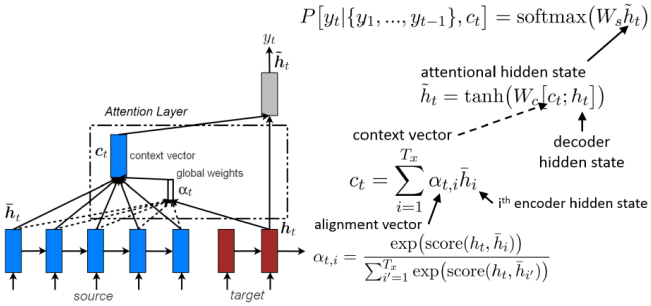
$$P[y_t|\{y_1, ..., y_{t-1}\}, c_t] = softmax(W_s\tilde{h}_t)$$

where $\tilde{h}_t$, the attention hidden state, is computed as :

$$\tilde{h}_t = tanh(_c[c_t; h_t])$$

h_t is thet^{th} hidden dtate of the decoder, c_t is the source context vector.

Figure 5. Summary of the global attention mechanism



5.4. Using a Transformer Decoder

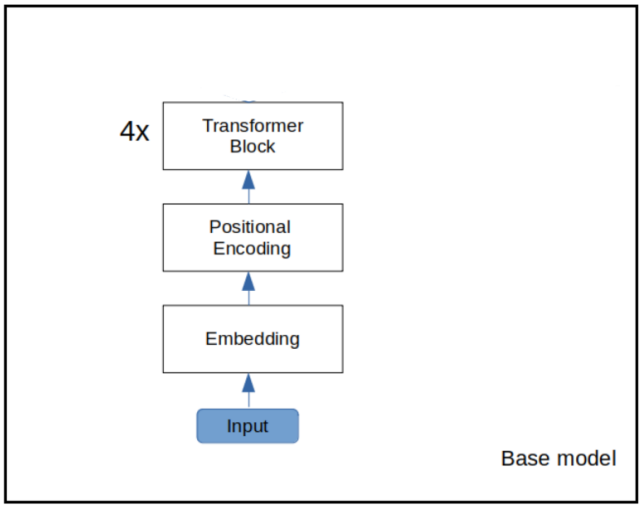
Our model is based on Transformers [7]. While the Transformer is a model that follows the encoder-decoder structure, it is possible to use only the encoder part (as in BERT) or the decoder part (as in gpt) to perform some specific tasks. In our task, we use a multi-layer Transformer decoder for generating Music. Fortunatly, PyTorch recent releases include a standard transformer blocks that can be easily used and adapted.

We use the square mask in the transformer just before using the Softmax function in order not to take into account some values from inputs (we change these values to $-\infty$ so that $e^{-\infty} = 0$). In the music generation task for example, we don't use the inputs that are ahead of the current character since we want the predict the next characters using only previous ones.

5.5. Using GAN to generate music

Another representation to store piano notes is using the **piano-roll representation**. Motivated by traditional piano rolls, a piano-roll representation is understood to be a geometric visualization of the note information as specified

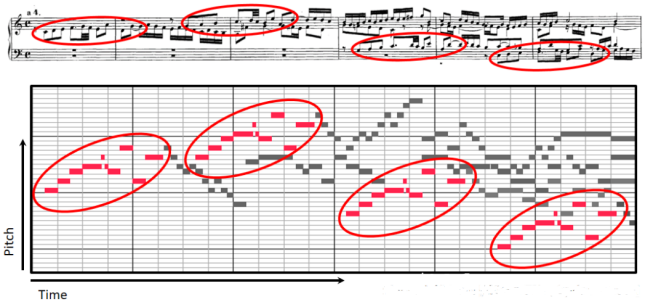
Figure 6. Transformer Decoder



by a piano roll. The horizontal axis of this two-dimensional representation encodes time, whereas the vertical axis encodes pitch.

The following figure shows a sheet music representation and as well as piano-roll representation.

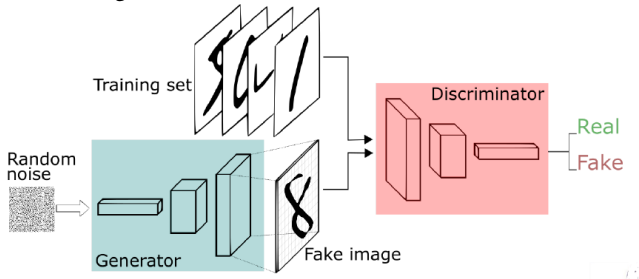
Figure 7. Piano roll notes



As we can see, we can represent a MIDI file as a black and white image where, at each moment, each black bar corresponds to a detected note. Hence, another way to generate music is to do the reverse; we start by generating the Black-and-white image and then transforming it into a MIDI file.

A well-known deep learning model to generate Black-and-white images (like MNIST digits) is **Generative Adversarial Network** [2]. This model ismade up of a first neural network, called the generator, that generates new data instances, and a second neural network, the discriminator, that evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not. For this model, we used a Dataset of Jazz Music Midi files which were transformed into piano roll representations.

Figure 8. Generative Adversarial Network



6. Results

6.1. BLEU score

BLEU[3] (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been generated by a natural language model. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics.

Scores are calculated for individual generated segments (generally sentences) by comparing them with a set of good quality reference generations. A perfect match results in a score of 1, whereas a perfect mismatch results in a score of 0.

BLEU is controled by the N-gram order : N. The BLUE is calculated by summing the N-gram matches for every hypothesis sentence S in the test corpus.

$$BLEU = \frac{\sum_{S \in C} \sum_{\text{N-gram} \in S} Count_{matched}(\text{N-gram})}{\sum_{S \in C} \sum_{\text{N-gram} \in S} Count(\text{N-gram})}$$

In our task, we compute BLEU for sequences of characters (ABC notation) instead of sentences. We do that using the **sentence_bleu** function already implemented in NLTK.

6.2. Results and Analysis

As previously indicated, BLEU was used to assign scores to the music generated by the NLP models. For the GAN model, we used the output of the discriminator as a score.

Results of each model

Method	BLEU score	Discriminator	Human rating
GRU	80%	-	6.5 ($\pm$3)
2-layer LSTM	71%	-	8.5 ($\pm$1)
RNN+attention	62%	-	7.5 ($\pm$1.7)
Transformer Decoder	57%	-	9 ($\pm$1)
GAN	-	0%	9 ($\pm$0.4)

As described in the abstract, we also added to those metrics a human evaluation, which basically consists of the grades between 0 and 10 that 20 volunteers decided to give for samples generated by each model. As we can see, even if the BLUE score varies from one method to another, the human rating consider the pieces of music as real. It's the same for the GAN, by listening to a generated song, we have the impression of listening to real music. But the discriminator score is unreliable because a score of 0 does not really reflect the quality of the music, it just means that the discriminator has been well trained to detect real music.

7. Conclusion

In this project, we looked at how to generate music with multiple Deep Learning Techniques. At a first time, we mainly relied on the latest advances in the NLP field by representing music as a sequence of characters using the ABC notation.We tested four different approaches that have all been shown to be effective in generating text. The first two methods are mainly based on the use of RNNs (GRU and LSTM). In the other two methods, we tested the effectiveness of the attention mechanism in the field of music. We started by implementing the many-to-many RNN with the attention mechanism. We then tested the Transformer Decoder which is made up of a multi-ahead attention with square mask. All these approaches need a song start so that they generate the rest of the song, that's why we looked for a new model capable of generating music without its start. The last model uses another representation : the piano roll which can represent a music play by a black and white image. We made the analogy with the MNIST digits generation, that's why our choice was GAN.

The most challenging task in music generation is evaluating the musical output. Any generated piece of music can generally only be subjectively evaluated by human listeners. Good music is easily recognizable by humans, so we could try to have a qualitative evaluation of the generated music. However, we tried to use also quantitative evaluations that allows to calculate to what percentage a generated music looks like real music. For the NLP models, we used the BLEU score already implemented in NLTK, and the the GAN model, we used the discriminator neural network that output the probability that a generated music is real. In terms of results, almost all the methods produced music similar to real ones. But if the choice of models for the metric scores has been successful, the choice of metrics can be improved.

References

[1] R. W. Allen Huang. Deep learning for music.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. June 2014.

[3] M. Post. A call for clarity in reporting bleu scores. In *WMT*, 2018.

[4] A. S. B. K. Rachel Manzelli, Vijay Thakkar. Conditioning deep generative raw audio models for structured automatic music.

[5] D. Rubino. Nottingham music database.

[6] A. van den Oord et al. Wavenet: A generative model for raw audio.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. 2017.