DeepClassic: Music Generation with Neural Neural Networks

Stanford CS224N Custom Project

Raphael Abbou
Computational and Mathematical Engineering
Stanford University
abbou@stanford.edu

Abstract

Art generation is an exciting research field in the domain of AI. Music is a very structured, but extremely complex, form of language, but when thinking about composing, one would primarily think that it comes to human genius, creativity and emotions, and disrupting music codes has always been the aim of the successive method of composition through musical history. It is thus extremely interesting to be able to try out new models to see how innovative they can be for music composition tasks. In this paper, we are interested in finding if sequential Deep Learning Models can be as efficient in music generation as they are in Natural Language Processing. We develop RNN, LSTM and LSTM with attention models, we manage to create short music scores that actually sounds like it could be created by a composer.

1 Introduction

Our aim is to design a network that could automatically generate piano music. The network is trained based on a predictive task: we fit the network to predict the representation of the note to come based on the previous notes that have been played. We then use this predictive model to generate notes that are likely to appear in a human-composed piece.

Furthermore, studying the relationships between time-dependent events in music, and learning how to find automatically patterns and structures in music tracks through Deep Learning approaches is probably useful in other domains like finance. A particularly interesting bridge between the two fields is that piano recording analysis has to deal with different time scales [1], to understand links between events happening at the scale of few milliseconds, to broad relationships on the scale of few seconds of minutes, which create a musical phrase or a movement. We believe similar links exist in Finance, as studying markets means both dealing with structural events at the scale of the day, month or year, and also studying events that happens every millisecond in High-Frequency-Trading.

State of the art models in this domain use more complex structure [1] than the ones we develop here, but in this approach we design simpler Deep Learning approaches that are similar to recent one [2]. The originality of our work comes from the fact that we use a particularly impressive dataset (which we will describe later) that has been released in 2019. One of the purpose of this work was also self-learning, which is why the code base is mostly implemented from scratch. One of the main challenges of this problem was to handle the data, and go from midi files to numerical data that could be used as input in a Neural Network, and that are good representations of the information carried out by music notes.

2 Related Work

We base our research on models that were state of the art few years ago [2], in order to be able to implement most of those parts ourselves and almost from scratch, for learning purpose, and because

we have limited time. In comparison with our model, state of the art research in this domain [1] uses a different music representation based on WaveNets [3] which are vectors representing waves produced by music instead of a pitch representation like the one we use. They also use transformers, while we implement sequential models, for music prediction and generation.

However, our model deals with more complexity in the representation than more basic ones, as we take into account both pitch and duration in notes and not pitch only, which allow us to produce complex melodies and rhythmics.

3 Approach

3.1 Note Representation

Our aim is to generate a model that can create piano music that sounds like it has been written by a human composer. First, we have to generate inputs that we can feed into our neural networks. For that, we rely on the music21 library [4].We use some programs implemented in the musicautobot project [5] to go from midi files to a note/duration representation, but all the programs related to the models and the formatting of inputs for those models is ours. We are able to transform a midi file into a tensor that describes the pitch of the notes and their duration. For example, let's consider the following chords and notes:

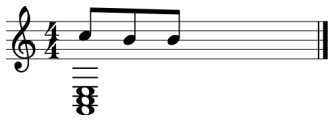

Figure 1: Music Example

A time unit corresponds to one 16^{th} of a a time step (here we have 4 steps per bar). In our representation, each note for a given time will be given by its pitch, followed by its duration. A separation token 'xxsep' allows us to know how to separate notes that are played together from the notes that are played afterwards, and how long we have to wait before the next note. The above notes are represented by the following list:

$$[xxbos, xxpad, n72, d2, n52, d16, n48, d16, n45, d16, xxsep, d2, n71, d2, xxsep, d2, n71, d2]$$

where 'xxbos' is the start token, 'xxpad' is the padding token, the first chord is represented by 'n72 d2 n52 d16 n48 d16 n45 d16', i.e. 4 notes, three of which have duration d16 ($16\frac{1}{16}$ =one full bar), and one which is a 16^{th} note 'n72 d2'. 'xxsep d2' tells us that we have to wait an additional 16^{th} time before playing the next note, 'n71 d2'. We then transform those vectors into one-hot vectors.

3.2 First Model: A Simple RNN

Our first approach consists in creating Recurrent Neural Network able to predict well the next note/duration given the previous notes and durations. We thus train by batches our RNN, inputting the on-hot representation of our notes, to predict the same sequence shifted by one. We add an additional linear layer to the hidden states, before passing them through a filter and then a softmax, to get a probability distribution over the note embedding:

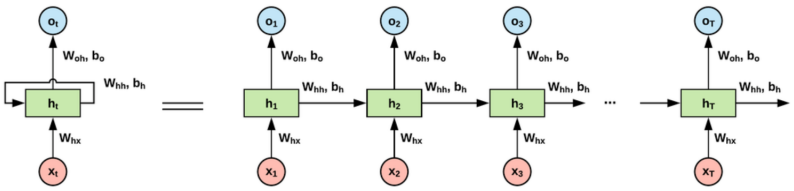

Figure 2: Music Prediction RNN

One specificity of our note representation is that each note is represented by a tuple (pitch/duration). To be able to retrieve a midi file from our generated output vectors, we have to enforce an output with

the correct format, i.e. where each note/separation token is followed by a duration, and vice versa. We therefore applying a filter to the output o_t before taking the softmax over the vector:

$$f(o_t^{[i]}) = \begin{cases} -\infty, & \text{if i in same range as previous range} \\ o_t^{[i]} & \text{otherwise} \end{cases}$$

Where the ranges can be whether the indexes of tokens related to duration or related to pitch. We will use this filter on the outputs o_t of the RNN before taking the softmax, so that the probability of choosing a duration (resp. note/separation token) if the previous token was a duration (resp. note/separation token) is null.

3.3 Second Model: LSTM

The limit of the simple RNN is that most of the information that is fed forward is about the last few hidden layers, and the backpropagation will be difficult for earlier layers because of vanisging gradient problems. However, in music, notes that are at the beginning of a musical sentence might be more important as they can give the ton of the subsequent musical part, among other things. In our second model, we use a LSTM instead of a simple RNN, to be able to achieve better transmission of information between the different layers of the Neural Network. Indeed, as it will be discussed in more details in the result section, the RNN manages to generate coherent results in a very short period of time, but then generates notes that seems more and more random. The LSTM is supposed to account more for that problem.

For music generation, at each step, given notes that have been generated already, we sample a new note from the last softmax output (here, we use sample instead of armgax in order to have more originality in our generated music), and take this generated note as the input of the next node:

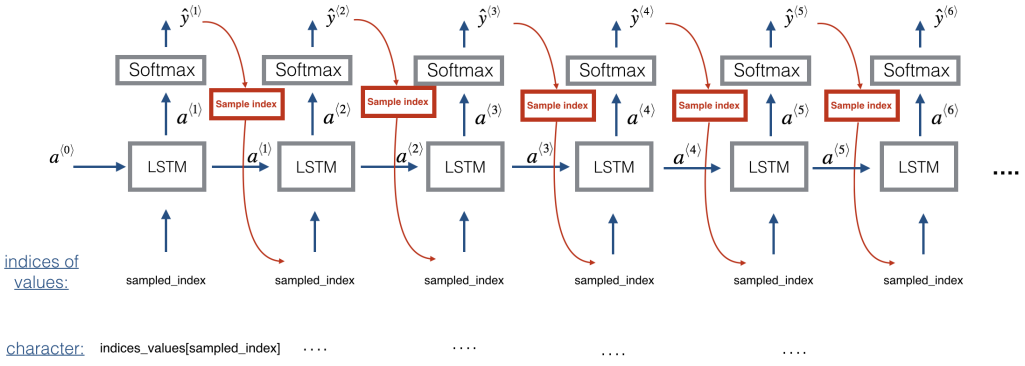

Figure 3: Music Generation Framework with LSTM

3.4 Third Model: LSTM with Attention

Finally, in order to help our model transfer information for earlier notes and generate more coherent musical pieces over longer time period, we add a feed-forward attention layer to our LSTM, based on the research paper by Colin Raffel [6]:

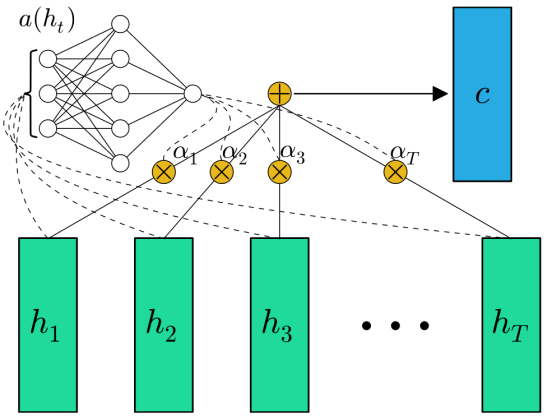

Figure 4: Attention Mechanism

With the following:

$$
\begin{aligned}
e_t &= W_a h_t + b_a \\
\alpha_t &= \frac{e_t}{\sum_{k=1}^{T} exp(e_k)} \\
c &= \sum_{t=1}^{T} \alpha_t h_t
\end{aligned}
$$

This attention mechanism allows us to focus on the most important previous hidden states to generate the next state by concatenating $[c_t, h_t]$, which is then, as previously, going into a linear decoder layer, a filter and a softmax.

4 Experiments

4.1 Data

The MAESTRO dataset that we take from Google's Magenta project [1] contains 172 hours (103GB) of piano recordings by virtuoso pianists who played during the International Piano-e-Competition, in MIDI files. The data has been gathered through 1184 performances, and contains more than 6 millions notes in total. We use 10% of the dataset as a test set, taken randomly for all samples. The samples are mostly from composers between the 17th and 20th century, like Chopin, Liszt, Beethoven...

4.2 Evaluation method

To compare our different model, we compute the precision, recall and F1 score for each of them, on both train and test set. Here, we define:

$$
\begin{aligned}
Precision &= \frac{1}{|C|} \sum_{c \in C} Precision(c) \\
Recall &= \frac{1}{|C|} \sum_{c \in C} Recall(c) \\
F1 &= 2 * recall * precision / (recall + precision)
\end{aligned}
$$

Where C is the set of note/duration classes, and with, for each c:

$$Precision(c) = \frac{\#TruePositive}{\#PredictedPositive}$$
$$Recall(c) = \frac{\#TruePositive}{\#ConditionPositive}$$

4.3 Experimental details

We fine-tune our hyperparameters on a sub-sample of the Maestro dataset, in order to be able to iterate fast enough. For the RNN, we end up with a size of hidden layer of 512, a learning rate of 0.001, number of epochs of 1000, batch size of 64. We keep the same hyperparameters for the LSTM.

For the attention mechanism, we use an arbitrary (we did not fine tuned those parameters) size $512 * 512$ for the matrix W_a, in order to have a length consistent with hidden layers.

4.4 Results

We get the following scores for each of the three models that we have implemented:

	Train			Test		
	Precision	Recall	F1	Precision	Recall	F1
RNN	60.2	59.7	59.9	58.9	57.3	58.1
LSTM	67.5	64.4	65.9	66.6	62.3	64.4
Attention LSTM	68.1	65.5	66.8	67.1	63.0	65.0

Figure 5: Performance Comparison for the different models

We can see that LSTM with attention is the best model. However, when we try to generate new music piece, it fails at producing coherent music piece when no prior structure is given (i.e. when we only use the <START> token as input).

We thus decide to add a small sub-sample from an existing human composed track, and generate a sequence to that input. We manage to generate samples that sounds much better. We quantify that by asking to 14 people (we couldn't get more people involved in the desired time) to rate on a scale from 0 to 5 how close from a human based composition the samples that they listen to, one being generated by each model. This methodology is inspired by a a human-based metric used in Google's Magenta project [1]. We obtained the following results, where for each model, samples are generated without any prior input or with the beginning of a human-composed sample:

	Without Previous Notes			With Previous Notes			
	RNN	**LSTM**	**A-LSTM**	**RNN**	**LSTM**	**A-LSTM**	**Real Music**
Scores	1	2.5	2.6	2.2	3.5	3.8	4.9

Figure 6: Performance Comparison for the different models

We can notice that adding few notes (and thus an original structure to start with in the hidden input layer) to our input makes the ouput much better, and gets a way higher human-based score.

5 Analysis

Based on the research that we have carried on, we can notice that simple structures like RNN can catch the essence of what makes a music piece, but they struggle to generate coherent music structure on a longer period of time. LSTM and especially LSTM with attention mechanism manage to establish links between notes that are further apart, as intended by their structures.

If we compare with state-of-the-art results in that domain [1], we get lower F1 score on the training dataset. One first main improvement would be to go from a one-hot representation of our vector to a Word2Vec-like representation, or a WaveNet [3] representation.

The music that is generated also struggles, for generated sequences that are really long (1 minute), to have the same structure as a music sample from a human composer. Indeed, even if each sub-part of it sounds correct (sub-sample of 10 secondes), our model does not manage to reproduce pauses between melodic sentences, or subtle changes in harmony. We believe that this is mainly due to the weakness of our representation more than the model (LSTM) itself. Indeed, a vector representation would, among others, allow strong similarity between notes separated by one octave to be similar, and to make notes within a same harmony closer. In addition of making our overall representation better, this would make our attention model more efficient and thus allow a better long-term coherence of our generated sample.

6 Conclusion

We manage to build different model with constant improvements in our music generation ability: our final model manages to generate interesting music sheets if we allows it to predict notes to come based on a small sub-sample of an existing music piece. Among others, we experienced how important memory mechanism were in order to create long-term structure and coherence, and that attention was in this situation, as it is in Neural Machine Translation, a pertinent mechanism to keep information from earlier layers.

In our research, we spent a lot of time manipulating the raw data and understanding large libraries in order to go from MIDI files to numerical inputs. We also had to deal with a lot of steps that were already done in assignements, like writing code for training, prediction, encoding, which was really interesting in order to be able to be able to implement Deep Learning models using the PyTorch library only.

References

[1] Curtis Hawthorne and al. Enabling factorized piano music modeling and generation with the maestro dataset. In *International Conference on Learning Representations (ICLR)*, 2019.

[2] Cheng-Zhi Anna Huang and al. An improved relative self-attention mech-anism for transformer with application to music generation. In *arXiv*, 2018.

[3] Aaron Van den Oord and al. Wavenet: A generative modelfor raw audio. In *SSW*, 2016.

[4] music21: a toolkit for computer aided musicology. `https://web.mit.edu/music21/`.

[5] Musicautobot. `https://github.com/bearpelican/musicautobot`.

[6] Raffel Colin and al. Feed forward networks with attention can solve some long-term memory problems. In *arXiv*, 2016.