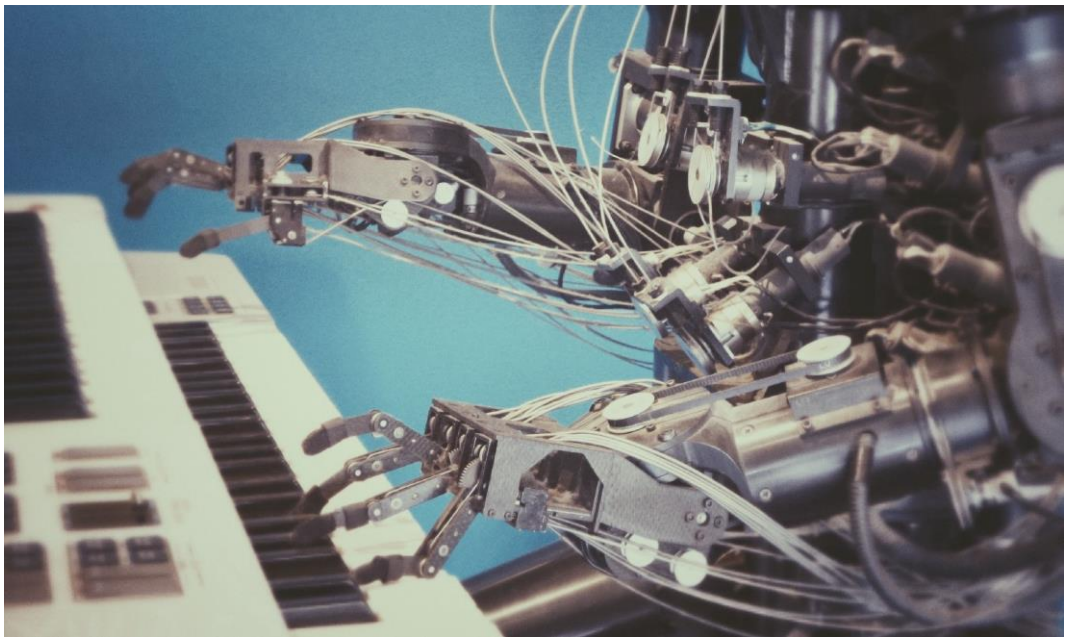
Music Generation Using Deep Learning

Machine Learning 2 (097209)
Spring 2021



Guy Lorinczi 311153845

Ran Brailovsky 308570928

abstract

In this project, our goal was to be able to generate music out of a seed sample, based on existing music files which the model will be trained on.
The idea behind it, is that music as we know it consists of a certain structure that follows specific theoretical music concepts which make it enjoyable to us. This structure can be learned by exposing a model to many music samples, which enables the model to predict which notes are more likely to follow the previous ones in the sequence. As a result, the model is generating new music.
Since the model learns the structure of a sequence, we used an RNN (Recurrent Neural Network) to fit the data, we experimented with a few different kinds of LSTM's (Long-Short-Term-Memory) and achieved the best results using Stacked LSTM architecture.

Introduction

Music has been for thousands of years a source of great pleasure for the humankind, but for most of human history, it has been created only by humans, in various ways. For decades, researchers had been trying to create a machine which generates music without the creativity of the human mind. Recent years have brought great prosperity to computer generation methods in many fields, probably most prominent of them is the field of image generation. Using these new techniques, researchers have been able generate amazingly realistic images, especially of human faces [1]. Video and voice generation have seen great improvements too, as can be seen in the greatly hyped 'deep fake' industry ([2] ,[3]) which has emerged based upon them. Images are very fast and easy to grasp by us humans, making the result of image generation easier to grasp and be amazed by, while videos and voice generation have enormous demand. Music generation exists and had major developments as well, but has seen less publicity, in our opinion, due to the lack of public interest in it and the fact that it requires more attention in order to be perceived.

In this project, we take advantage of a well-known model called LSTM [4] and use several different architectures based on it, in order to tackle the problem of generating music. At first, as the base part, we will review a simple model architecture that consists of a single LSTM layer and a single output layer. After discussing the results and conclusion of this part, we will review both Bi-LSTM, Hybrid LSTM and Stacked LSTM, some of which will show great improvements over the base model, and some won't. We will then offer some more research clues which we suspect could lead to even better results in the future.

Data and preprocessing

Out of a very large MIDI files data set called "Musical AI MIDI Dataset" we focused on classical piano music and used 86 different original music pieces composed by famous classical composers such as Beethoven, Bach and Mozzart [5].

We decided to use MIDI files as the input to our model and generate MIDI files as well as their structure enables easier processing instead of using audio files directly.

Each file consists of several instruments from a wide variety of 128 unique instrument kinds. Each instrument has a list of notes, and each note is described by its pitch, velocity, and the time it plays. In fact, the music in the MIDI file is given by stating a defined set of instructions, which if played according to the format, creates the music as we can hear it, in an audio format. There are multiple existing libraries which enable converting MIDI files into easy-to-work-with objects, and those objects usually have a textual representation of those instructions.

The textual representation of the music allows easier understanding of the music saved in the file and enables better manipulation of the data for the processing stage. The MIDI format also saves quite a bit of training time since it is much lighter (in file size) than its standard audio file counterparts.

In order to work with the data, we went through several more steps before feeding it to the model. The first step after having the MIDI file converted into the set of notes instructions, is to translate those instructions into a format called piano roll (figure 1). The piano roll is actually a two-dimensional list of time and pitch, so that each column represents a different point in time and the rows are the different possible pitches (out of 128) which play at that given time. The time intervals which define the piano roll are derived by the sampling frequency parameter: each column is spaced apart by 1/sampling frequency seconds.

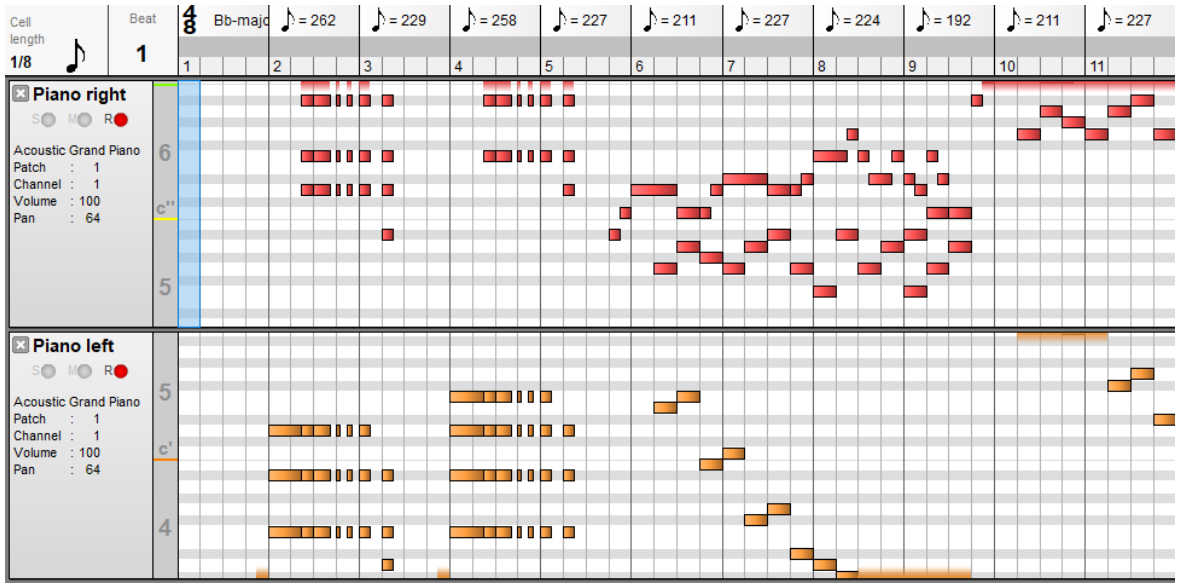


Figure 1- MIDI file processed and viewed as piano roll

The resulting piano roll is a very sparse matrix which holds many zero values for all the notes which aren't played, at each time unit. This calls for a feature extraction that can drastically reduce the dimension of our data. To do so, we used a tokenizer which creates a token for each unique combination of note pitches. After doing so, we can translate the piano roll into a much denser representation as a list of integer values, where each value is a single token.

When looking at the data from a time sequence point of view, we wanted to split the data into an input list and a target list where each sequence of notes in the input corresponds to a single target value which is actually just the next note in the sequence. We used a fixed size window which represents the input sequence length and used the sliding window method in order to split each song into several windows. For each window, we created the target note and represented it as a one-hot vector of the different note classes from the Tokenizer.

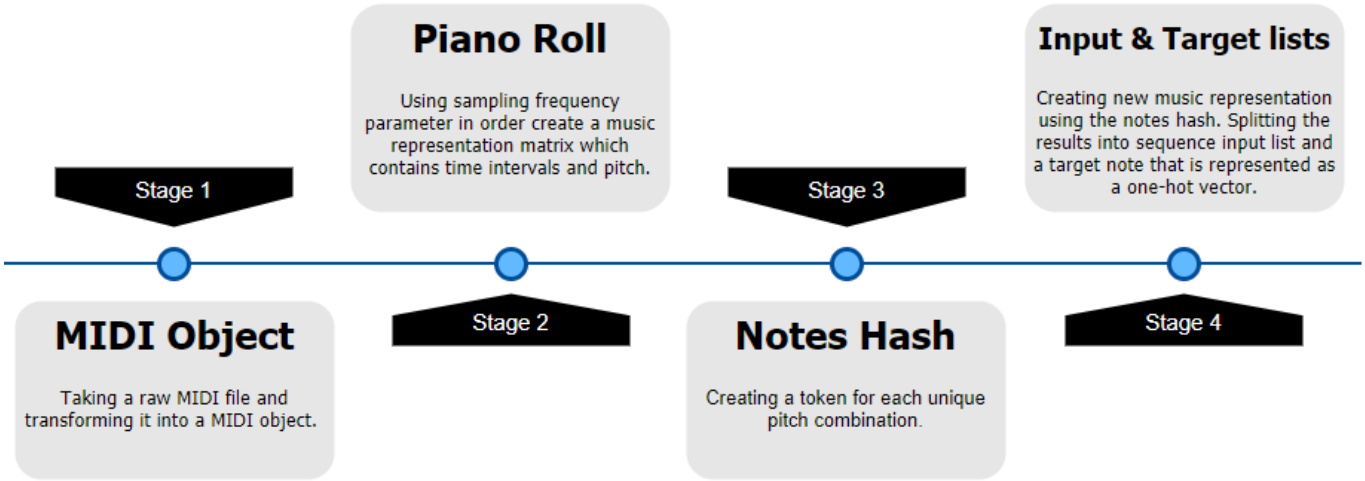


Figure 2- Preprocessing stages

At the end of the preprocessing of each MIDI file, we have an input list that contains all the window sized sequences and a target list with the corresponding target note, represented by a one-hot vector.

Basic Part

LSTM- Baseline Model

Since the task is obviously based on sequential data, RNN's are a trivial fit which enables learning a structure out of a time series. More specifically, we used one LSTM layer for the baseline model as it seems to be well equipped for learning a continuous theme along the music sample by embedding the previously learned structure inside its inner hidden state [4]. Besides the regular LSTM model hyper-parameters, we had to find the optimal **window size** and **sampling frequency**- two parameters that which rely on our preprocessing and had a great impact on the generated song as we have seen in our experiments.

At first, we decided to preprocess a single song, and try to re-write the corresponding MIDI file from our processed notes representation. This way, we could compare the two MIDI files: the original one and the one reconstructed after the preprocessing phase. When doing so, we noticed that initializing the sampling frequency with low values results in a generated MIDI file that is missing several notes, especially when the song is originally played at a high tempo. Increasing the sampling frequency resulted in a much more accurate reconstruction. The window size parameter has a tight connection to the sampling frequency as if the sampling frequency increases, so does the window size need to increase in order to be able to "catch" the structure of the song correctly and update the LSTM hidden states accordingly. We saw that increasing both parameters allowed us to process more relevant data, but also increased the training time significantly. We saw that even though the baseline model has generated music, it did not make much sense to the human ear. This led us to further investigation of more complex sequential models that will do a better job of learning the underlying structure of the music as we, humans, enjoy.

Advanced Part

Bi-LSTM

After evaluating the base model of LSTM and discussing its feed-forward limitations, we concluded that the structure of music might have a "reversed" connection: instead of having a specific note dependent only on the notes that were played beforehand, we saw a potential in catching also the connection between the note and the notes that are played after. This kind of "bidirectional" dependency led us to use a different kind of LSTM model: Bi-LSTM [6]. Bi-LSTM model considers and evaluates the relationship between notes sequence in both forward and backward direction; this bi-directional approach can allow us to catch inner sequence connections that could not be discovered by the conventional LSTM [7].

Stacked LSTM

When discussing a way of adding deeper perception to our sequential model, apart from adding a simple RNN layer to increase the network's depth, we came across research that proposed a model that utilizes the advantages of LSTM in capturing relevant connections through time and making it more robust by stacking over it another LSTM layer [8]. Adding another LSTM layer to the model instead of adding more hidden fully connected layers on top of it, preserves the sequential properties of the data while allowing more freedom and complexity to our model. Adding more depth and complexity can improve the model's accuracy, resulting in notes generation that better resembles real music.

Creative Part

Hybrid LSTM

For the creative part, we decided to create and utilize a convolution layer that is generally used in image processing in order to extract features and reduce the input dimensionality even more. Using both CNN and LSTM is not a very common scenario but we've found some honorable mentions of it being used for similar tasks ([9], [10], [11]) and decided it was worth a try. We added a convolution layer of a single dimension between the input and the LSTM layer. Doing so enables the convolution layer to extract the relevant features from the entire sample before feeding only the more relevant structure of the time-based sequence to the LSTM layer. Moreover, we decided to add a time-distributed layer that is commonly used in one-to-many and many-to-many sequential models, enabling the model to better capture the relationships between individual LSTM cell values at each timestamp. Doing these adjustments resulted in a faster training procedure while also producing pretty good generation music results.

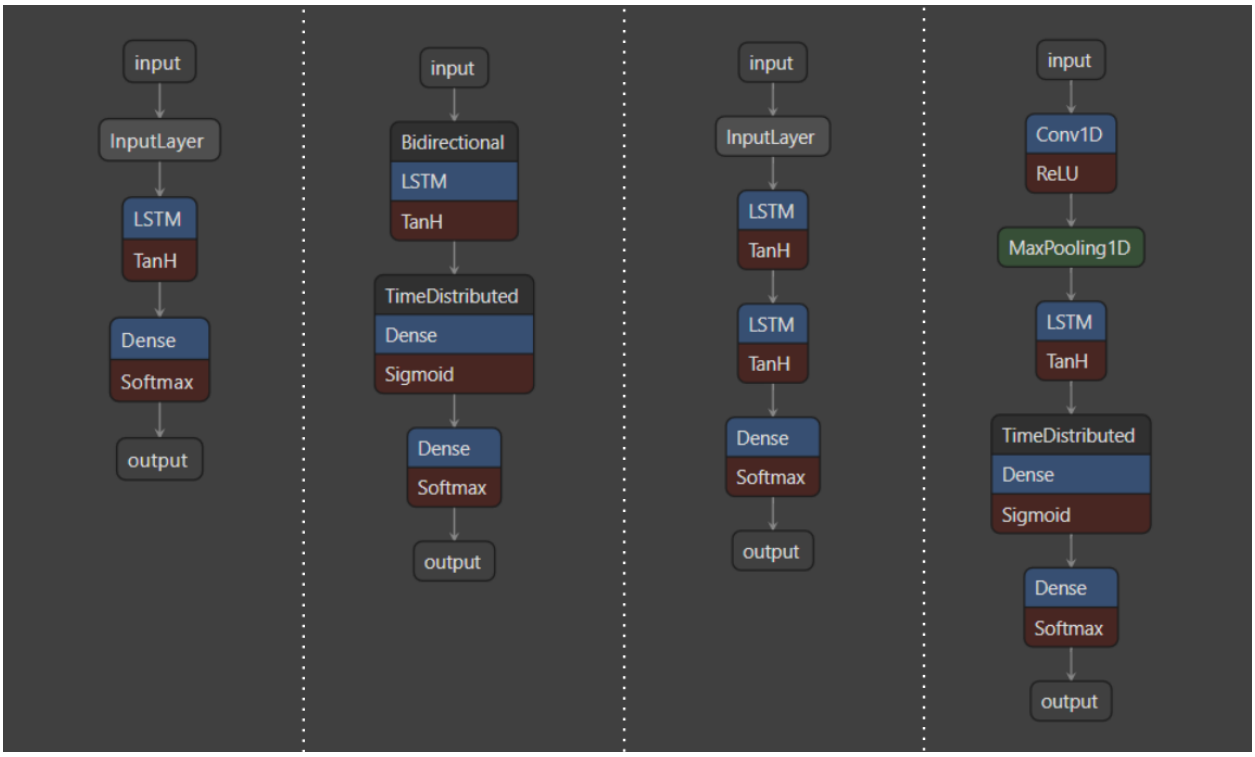


Figure 3 - Models Architecture: LSTM, Bi-LSTM, Stacked LSTM & Hybrid LSTM

Results and Conclusions

Comparing all the described models (figure 3) by their empirical measures as loss values or accuracy could be misleading since those only refer to the difference between the model predicted notes and the actual notes of the real songs. Also, since some models were much more time consuming to train, and some showed no further progress by training, having a benchmark of training for a certain number of epochs of time seems irrelevant. We trained each model until it reached quite low loss values or until it seemed like it has reached a convergence and probably won't improve much more. Evaluation of generated music pieces and computer-generated materials in general seems to be elusive and quite challenging [12]. At large, this problem can be tackled in two main approaches: Subjective methods and Objective methods [13]. The subjective approach is based upon human decision making, evaluating the difference pieces one by one while being requested to follow a certain rule-based guideline. The objective approach relies on statistical measures instead of humans. This may sound preferable for both resources scalability reasons and accuracy reasons as well, but it is actually not always the go-to solution, since the process of statistically describing what sounds better to the human ear can be quite hard to do. In our comparisons, we first rushed to listen to each generated piece and evaluate it ourselves, implementing the subjective approach. Trying to use the variation of the Turing test described in the paper of Yang, L. C., & Lerch, A. quickly revealed itself to be insufficient for this cause for the simple reason that almost all the generated samples were easily recognizable as machine generated and not real classical music samples. We therefore used a subjective scale to differentiate between the generated pieces by ordering them by their similarity to the original classical music in the dataset. Later on, we decided on evaluating the different models in a more objective way by comparing the distribution of the different notes and measuring its distance from the distribution of the total dataset. As demonstrated in *Figures 4-6* below, there is a major difference in the similarity of the generated notes distributions between the two models and the original notes distribution of the dataset. We found a significant correlation between the ranking of our subjective results and the ranking of the models based on their similarity to the original song's notes distribution.

We saw a significant improvement of the generated results with the higher complexity models, and this of course could be further extended for future projects. In particular we think a combination of the hybrid model and the stacked LSTM model could be interesting. We have also found that having a better domain knowledge, and a better understanding of music representation and even MIDI file's structure is beneficial for the success of such projects. Future continuation of the subject would greatly benefit from learning more data which exists in the MIDI file, like velocity of the notes and a dynamic tempo which changes along the song.

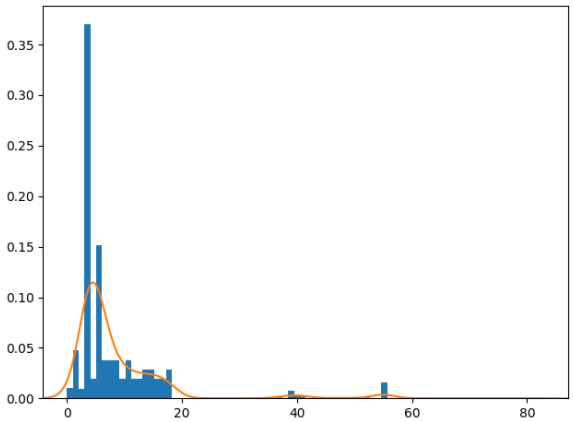


Figure 4- Baseline LSTM generated notes distribution

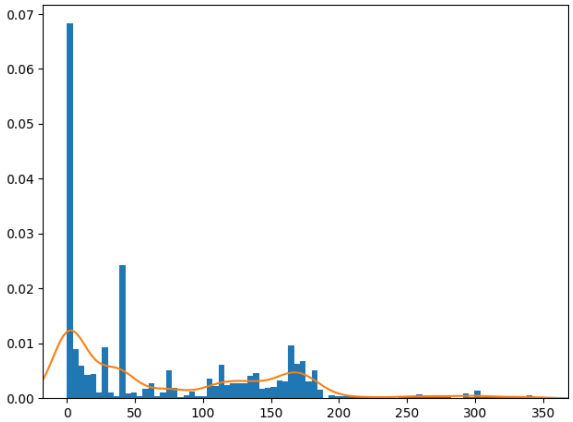


Figure 5- Stacked LSTM generated notes distribution

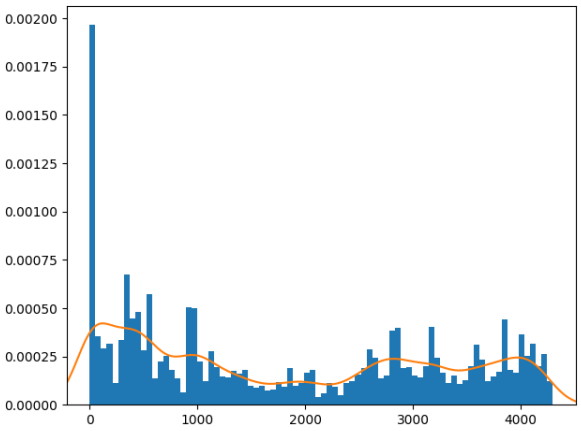


Figure 6- Original songs notes distribution

References

[1] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8110-8119).

[2] Arik, S. O., Chen, J., Peng, K., Ping, W., & Zhou, Y. (2018). Neural voice cloning with a few samples. *arXiv preprint arXiv: 1802.06006*.

[3] Song, L., Wu, W., Qian, C., He, R., & Loy, C. C. (2020). Everybody's talkin': Let me talk as you want. *arXiv preprint arXiv: 2001.05201*.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[5] Krueger, B. (2004). *Classical Piano MIDI Page*. Classical Piano MIDI. http://www.piano-midi.de/midi_files.htm

[6] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), 602-610.

[7] Jiang, T., Xiao, Q., & Yin, X. (2019, May). Music generation using bidirectional recurrent network. In *2019 IEEE 2nd International Conference on Electronics Technology (ICET)* (pp. 564-569). IEEE.

[8] Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. *arXiv preprint arXiv: 1801.02143*.

[9] Tovar, M., Robles, M., & Rashid, F. (2020). PV Power Prediction, Using CNN-LSTM Hybrid Neural Network Model. Case of Study: Temixco-Morelos, México. *Energies*, *13*(24), 6512.

[10] Fathi, O. (2019). Time series forecasting using a hybrid ARIMA and LSTM model. *Velvet Consulting*.

[11] Huang, Y., Huang, X., & Cai, Q. (2018). Music Generation Based on Convolution-LSTM. *Comput. Inf. Sci.*, *11*(3), 50-56.

[12] Theis, L., Oord, A. V. D., & Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv: 1511.01844*.

[13] Yang, L. C., & Lerch, A. (2020). On the evaluation of generative models in music. *Neural Computing and Applications*, *32*(9), 4773-4784.

[14] Project's GitHub Repository:
https://github.com/Ran-br/Music-Generation-Using-Deep-Learning