Composer AI

Generating Music using Deep Learning

Authors: Fazekas Gergő and Joseph Tadrous

Introduction

In this project, we aim to use neural networks for automatic music generation. Music generation is a famous topic nowadays and can be used in many applications. Musicians or artists build on what is generated by the machine and produce their own original work. The music or art generated by software is also sometimes sold by the companies or individuals who designed them. Since music is sequential data, it can be modelled using a sequential deep learning model as a recurrent neural network which is a member of feedback neural network. Therefore, a RNN combined with a convolutional neural network will be trained to learn the music sequence (input in midi format), and this will help in generating the sequence of music data.

Previous Solutions

One of the most popular composers AI out there is AIVA. AIVA was created in 2016, and it specializes in classical and symphonic music composition. Moreover, it's the world's first AI composer to be recognized by the music society SACEM. It has already released its own copyrighted albums filled with generated soundtracks in the electronic music genre (Travers, 2018). AIVA's algorithm is based on deep learning and reinforcement learning architectures. The exact architecture is confidential, and its details has not been open-sourced. AIVA has been trained on a large collection of existing works of classical music written by famous music composers as Mozart, Beethoven, and Bach. AIVA is capable of detecting regularities in music and on this base composing on its own. It is capable of generating short compositions in various styles (rock, pop, jazz, fantasy, shanty, tango, 20th century cinematic, modern cinematic, and Chinese).

https://www.siliconluxembourg.lu/aiva-the-artificial-intelligence-composing-classical-music/

https://www.aiva.ai/

https://medium.com/@aivatech/composing-the-music-of-the-future-4af560603988#.t8d6dkxi8

Dataset

Our dataset consists of classical piano midi files containing compositions of 19 famous composers. The dataset was retrieved and downloaded from Kaggle at https://www.kaggle.com/soumikrakshit/classical-music-midi.

Then a pandas dataframe was created from the midi files. Only the piano left and piano right instruments are used. The file is ignored if it does not contain the aforementioned instruments. The dataframe contained the note, its pitches, velocity, and duration.

		pitches	velocity	duration
	0	[72]	40.000000	0.251446
	1	[65, 69, 77]	35.666667	0.127866
haydn/haydn_9_3.mid	2	[72]	40.000000	0.127866
	3	[77]	41.000000	0.127866
	4	[72]	42.000000	0.127866
...	...	...	...	...
	3105	[44, 48, 51, 56, 75, 80, 84, 87]	77.375000	0.277136
	3106	[37, 41, 44, 49, 77, 80, 85, 89]	77.375000	0.277136
liszt/liz_rhap12.mid	3107	[32, 36, 39, 44, 80, 84, 87, 92]	77.375000	0.277137
	3108	[37, 41, 44, 49, 85, 89, 92, 97]	77.375000	1.078652
	3109	[54, 58, 61, 66, 73, 78, 82, 85]	80.875000	1.136797

357130 rows × 3 columns

In the data preprocess phase, we converted the pitches to multi-hot encoding and windowed the sequence, while the velocity and duration columns were then scaled using the MinMaxScaler() method.

Proposed Method

Our approach includes creating and training two different models, and combining them for the results. Since pitches are encoded using multi-hot encoding (classification problem), our first model is a CNN trained to handle pitches input. Its architecture consists of 3 Conv1D layers, 2 Dense layers, and a dropout. Since duration and velocity are continuous values (regression problem), our second model is a RNN (LSTM) model trained to handle the duration and velocity inputs. The architecture of the second model consists of one LSTM layer, 2 Dense layers, and a dropout.

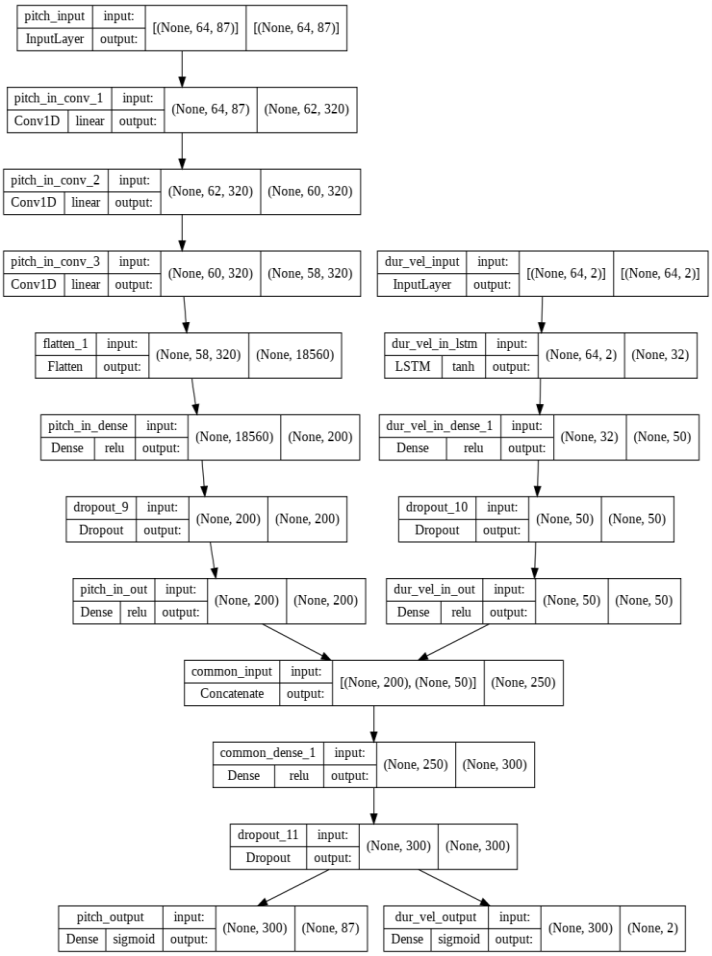
Layer (type)	Output Shape	Param #
pitch_input (InputLayer)	[(None, 64, 87)]	0
pitch_in_conv_1 (Conv1D)	(None, 62, 320)	83840
pitch_in_conv_2 (Conv1D)	(None, 60, 320)	307520
pitch_in_conv_3 (Conv1D)	(None, 58, 320)	307520
flatten_1 (Flatten)	(None, 18560)	0
pitch_in_dense (Dense)	(None, 200)	3712200
dropout_9 (Dropout)	(None, 200)	0
pitch_in_out (Dense)	(None, 200)	40200

Total params: 4,451,280
Trainable params: 4,451,280
Non-trainable params: 0

Layer (type)	Output Shape	Param #
dur_vel_input (InputLayer)	[(None, 64, 2)]	0
dur_vel_in_lstm (LSTM)	(None, 32)	4480
dur_vel_in_dense_1 (Dense)	(None, 50)	1650
dropout_10 (Dropout)	(None, 50)	0
dur_vel_in_out (Dense)	(None, 50)	2550

Total params: 8,680
Trainable params: 8,680
Non-trainable params: 0

Our combined and final model can be summarized in the following figure:



As for the loss, we added much bigger weight for the classification of the pitches, because it is much more important to pick the proper piano key, then the velocity, or duration values.

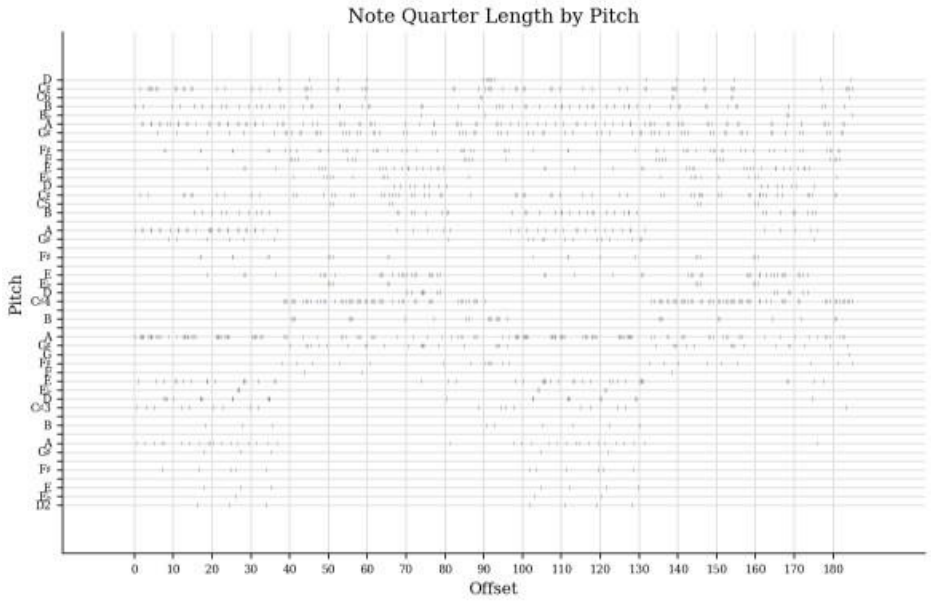
Evaluation Method

An objective and subjective evaluations were carried out to evaluate our model. For the objective evaluation, the loss, pitches output loss, duration & velocity output loss, and pitch output accuracy were computed.

For the subjective evaluation, our goal is to generate music that is listenable and cohesive. Thus, the Music Generator Class was created. This class generates music using the fit model and the scaler. The temperature parameter introduces randomity to the output. It can be changed to avoid stucking into one simple pattern (e.g. pressing one piano key endlessly).

Results and Discussion

As a first step, our combined deep learning model managed to learn the pattern of one of Mozart's pieces. We were able to generate midi files and play them. Even if we started the generation from a random state, it quickly found the pattern of the famous piece. We achieved on this small data around 70% accuracy for the pitches. Despite of the poor validation set results (30% accuracy); the generated music file was really realistic (in terms of the rhythm as well). It even successfully managed to learn the difficult part of the song.



However, we found much more challenging to make the model to learn universal patterns from plenty of music pieces and use that knowledge to generate a new song. At first, due to the RAM memory restrictions of the Google Colab workspace, we were not able to train the model on the entire dataset. The second difficulty was the required training time. Based on the previous results (learning only one song) around 500-1000 epoch were necessary to learn the more complex patterns and generate more listenable results. In case of bigger dataset, the required GPU time was much higher. And finally, we often faced with the issue, that the model tends stucking into one simple pattern, and if we increased the randomity (temperature parameter) of the generation, then the output became more arbitrary and less realistic. The best outputs were somewhere between the random pressing of the piano keys and what we call music.

Nevertheless, the result of our deep learning model was very satisfying (specially in case of learning only one song) based on the knowledge and experience we had.