An approach to Beethoven's 10th Symphony

Author Information
Anonymised for double-blind review

Abstract

Luidwig van Beethoven composed his symphonies between 1799 and 1825, when he was writing his Tenth symphony. As we dispose of a great amount of data belonging to his work, the purpose of this paper is to investigate the possibility of extracting patterns on his compositional model and generate what would have been his last symphony, the Tenth. A neural network model has been built based on the Long Short-Therm Memory (LSTM) neural networks. After training the model, the generated music has been analyzed by comparing the input data with the results, and establishing differences between the generated outputs based on the training data used to obtain them. The structure of the outputs strongly depends on the symphonies used to train the network, so the music obtained presents characteristics recognisable as a Beethoven-like style.

Introduction

Music is an art, but also a global language present in every historical stage. From the prehistory, to Medieval, Baroque or Classical, each stage has it is own social, politics, and also artistic characteristics, present in paintings, literature, and music.

Romantic composer Ludwig van Beethoven wrote his Symphonies from 1799 to 1825, when he finished the No. 9. Although there's no constancy of the existence of the 10th Symphony score, there exists some sheets found in Beethoven's house after his death that are thought to be part of the upcoming Symphony. Those sheets are kept in the museum dedicated to his life in his natal city, Bonn, although they can be seen online [1]. The public manuscript is not easy to read and understand, so that existing data will not be used in this paper.

In 1988 Barry Cooper, a musicologist who wrote a book relating Beethoven's life (Cooper 2000), built from 50 fragments, the first movement of the Symphony [2]. Since it can't be proved that those found sketches were intended to be part of the 10th symphony, Barry Cooper's work has caused a big controversy.

A legend arises from this cause, called *"the 10th Symphony curse"*. Following Beethoven's steps, several great composers were found death before finishing it is 10th Symphony. This is the case of authors such as Franz Schubert (1797-1828), Anton Bruckner (1824-1896), Antonín Dvořák (1842-1904) or Gustav Mahler (1860-1911). The last one tried to avoid the curse by not assigning a number to his ninth Symphony, in order to be able to assign the number 9 to his tenth Symphony. Despite his effort in avoiding the curse, he was found death while composing the last one.

The goal of this work is to generate music, based on Beethoven's compositional model, obtaining all the orchestra instrument's scores. The first approach has been to train the system with each instrument individually, to generate all the different scores, and then put them all together in a conductor's score. In contrast, the second approach has consisted in training the system with information from different instruments at the same time. The output is intended to be fully dependent of the system prediction, although the only characteristic that we have forced is the symphony's tempo and the key, as the sheets found in Beethoven's house after his death had 3 flats, and the tempo was a 6/8. He chose to write a large portion of his compositions in this key, as it is said that it represent a "stormy and heroic tonality", and it is used in works of unusual intensity, such as the Fifth Symphony (Fig. 1).

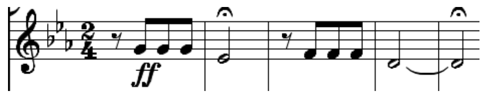

Figure 1: Snippet of Beethoven's Fifth Symphony in C minor

Regarding the C minor key, it was considered appropriate for masonic music to have that key signature, due to the importance that the number 3 and letter b had in the freemasonry. Relevant composers in the history such as Wolfgang Amadeus Mozart wrote music for masonic use, (Henry and Massin 2006). Although Beethoven is not documented as a mason, there are strong grounds for believing in that, since it is known that several of his compositions were played in the Masonic Lodge. This information is documented in *The age of Mozart and Beethoven* (Pestelli 1984).

Some musical definitions important to fully understand the paper development are:

[1] https://bit.ly/2BKPAOx

[2] https://bit.ly/2T5KBBH

- Note: Musical event that describes a sound. It contains more information apart from the note number, but also the duration, or the pitch class.

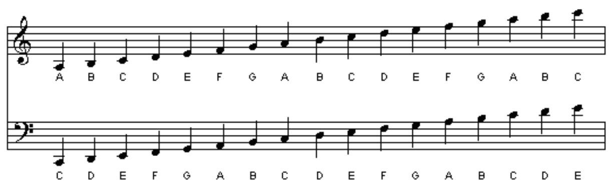
A B C D E F G A B C D E F G A B C
C D E F G A B C D E F G A B C D E

Figure 2: Notes names of the chromatic scale

- Pitch: Property of sounds that allows a frequency scale ordering, distinguishing between ”higher” or ”lower” sounds.
- Clef: Musical symbol used to determine the name and pitch of the written notes, it is the first symbol that appears in the score. The tree types are: F (second stave from Figure 2), C and G (first stave from Figure 2).
- Key signature: Set of sharp or flat symbols placed after the cleff, it determines the notes that will be altered from their natural pitch. A sharp raises one semitone the natural note, while the flat lowers it.
- Time signature: It determines how many beats are contained in each bar. It appears next to the key signature. As we can see in the example of Figure 1, the 2/4 time signature means that there are two crotchets or quarter notes per compass.

This paper is structured as follows. Previous work on using Artificial Intelligence in music generation is exposed in the State of the art section. After that, the work developed for this paper is explained in detail, presenting the Deep Learning technique used, the needed toolkits and how the data is represented. Then, the Music Generation subsection is divided in: Dataset description, training and prediction. The results section is focused on explaining the reason why the system returns a certain output when trained with a specific set of symphonies. The conclusions and future work are described in the last sections.

State of the art

Studied since the latter half of the 20th century, Computational Creativity can still be considered as a novel field. There has been relevant experiments on Linguistic creativity, such as narrative generation (Gervás et al. 2005), getting to write the lines of a musical called *Beyond the fence* (Gardner 2016), showed for the first time in London's Arts Theatre, but also poems (Montfort et al. 2012) or jokes generation (Ritchie 2009). Visual Arts creativity has been a recurrent conversation topic these last months due to the auctioned AI generated painting, *Portrait of Edmond de Belamy (2018)*, by the Christie's art gallery of New York, getting the price of 432.500$, whose algorithm was designed by *Obvious*[3]. Another relevant work on this field is AARON, (Cohen 1995), a robot capable of taking a brush with it is robotic arm and paint. The Painting Fool, (Colton 2012) emulates several styles.

[3]`http://obvious-art.com/index.html`

Music creativity

This Computational Creativity sub-field started in the early 50's, although the most relevant works are mainly focused on generating coherent sounds and scores for the human musicians use. The first used Artificial Intelligence technique for this purpose was the Markov chains. This model defines the probability for an event to happen based on previous ones, storing them in a transition matrix. An example of the application of the Markov Chains is *ILLIAC* (Hiller and Isaacson 1958). This machine generated the *ILLIAC's suite*, a string quartet [4]. Generated notes were tested by heuristic compositional rules. In case that the rules weren't violated, they were kept, otherwise a backtracking process was followed. This project excluded any emotional or expressive generation, by just focusing on the notes. Later on, a system called *CHORAL*, which produced the corresponding harmonization of a given Bach Choral, was developed creating rules and setting heuristics in a logic-programming language created by the author for this purpose (Ebcioglu 1990).

In 1989 a new technique was implemented for this purpose, Recurrent Neural Networks. Since that method turned out to be limited by it is short-term coherence, Long Short Term Memory (LSTM) neural networks started to be used for this purpose. This method incorporated the ability to learn long-term dependencies. Since music is built on themes and motifs repeating over time, it makes LSTM networks a reasonable option for computer music creativity. Melodies generated with LSTM networks in existing projects have resulted more musically plausible than with other models, such as Gated Recurrent Unit (GNR) (Nayebi and Vitelli 2015). The first music generation project that used neural networks is MUSACT (Bharucha 1992), which focuses on learning the harmonic model and generates expectations after listening to a certain chord. Some other projects that use various of this networks models to generate new sounds have been developed through the last few years, using raw audio (Kalingeri and Grandhe 2016). BachBot (Liang et al. 2017) composes and completes music in the style of Bach chorales using an LSTM generative model. They conducted a discrimination test to determine if the generated music was similar to Bach's chorales with 2336 participants, getting a rate of only a 1% of the people correctly determining which music was generated with BachBot.

Some other examples of music generation projects are *DeepMusic*[5], which is integrated in Amazon's assistant Alexa as a skill, so it plays AI generated music. Chinese company Huawei recently published the unfinished part, third and fourth movements, from Shubert's symphony No. 8 (Mantilla 2019), which the author left unfinished on purpose. Using neural networks, the system gave to the musicians some ideas to continue the music, and then musician and

[4]`https://www.youtube.com/watch?v=fojKZ1ymZlo`

[5]`https://amzn.to/2DBRwJc`

composer Lucas Cantor worked on them. The final version has been played on the 4th of February, 2019, in a unique concert in London.

Currently best-work known on computer music composition is EMI, (Cope and Mayer 1996). This system has successfully emulated Mozart, Brahams, Bach, Rachmaninoff or Chopin's music, generating new music [6]. It searches a pattern or signature as Cope's labeled, in at least two existing pieces of a concrete compositor. Using one of the artist scores, it locates the signatures to generate the new music, and in order to compose the music between signatures, it uses a rule analyzer. The IAMUS (Quintana et al. 2013), named this way after the god of music, Apolos's son in the ancient Greece, is a computer system created at *Universidad de Málaga*. It is capable of composing a full score in 8 minutes, using genetic algorithms, whose music has been played by the London Symphony Orchestra. In this case, chromosomes including all the notes information are randomly generated, and fitness functions are applied to each of them. If a note is codified to be played by a violin and this instrument doesn't have the possibility to play that note, it is changed. After generating around 100 scores, a human composer chooses the best one as the final output.

Another challenging field relating Musical Creativity is Music Improvisation, since it has more difficulties from a creative point of view. Using Genetic algorithms, GenJam (Biles 1994) emulates a Jazz musician in his or her improvisation learning process, while the Continuator (Pachet 2003) uses a Markov model to generate music in standalone mode, as continuations of musician's input, or as interactive improvisation.

Work description

Technical background

Deep learning: LSTM Networks Included in the field of Machine Learning, Deep Learning involves the use of neural networks in the task of providing *knowledge* to the machines. There exists several types of neural networks, such as Deep Neural, Deep Brief and Recurrent Neural Networks (RNN). In this paper we work with the last ones, since we need to process sequential data, assuming each event depends on previous ones. The most accurate RNN variant is LSTM. As proved with Figure 1, we need the memory that this type of networks own. We have the sequence F - F - F, a predictor without memory would return another F, although by learning from the notes before, it can extract that after three equal notes, it is probable that the upcoming note is two lines below the last one.

Proposed in 1997, those neural networks can learn long-term dependencies, improving the cells or neurons in the RNN graph. They have the ability to connect previous knowledge to a present task. Each cell has a certain memory, and it decides to store or forget a certain data based on a given priority, assigned by the algorithm after a certain time learning and represented as weights.

As it can be seen in Figure 3, the top line represents the flow of the cell state. Also, several layers are shown, such as

[6]https://bit.ly/2DDVKjF

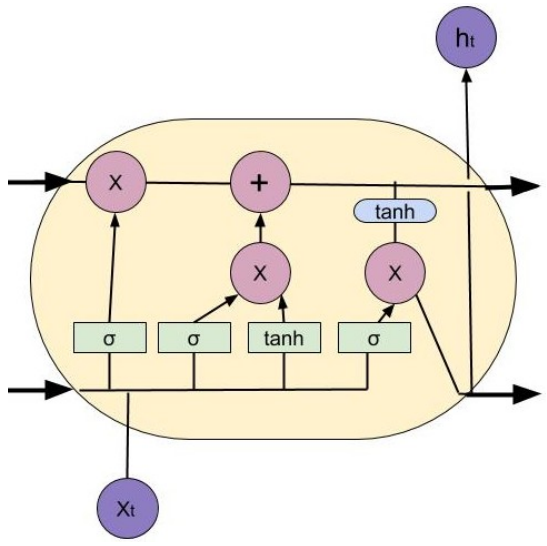

Figure 3: LSTM Neural Network cell

the first sigmoid, which takes information from the previous state and determines if it is useful or not, returning 0 or 1. As it is shown with the vertical arrow, it directly affects to the flow of the cell state. The second layer is composed of another sigmoid, which chooses the data to be updated from the previous state. The tanh component creates a vector of candidate values to be added to the state. The combination of both will be added to the current cell state. The final sigmoid layer decides which parts of the current state are more relevant. Those will be sent to a tanh function, which will convert the state into 1 or -1.

Toolkits

This project has been developed in Python, and the most relevant library used is Music21 [7], which allows parsing and generating scores in different formats. Also, every musical action and representation that we needed to perform, was made possible using that library. For the Deep Learning engine we have used Keras [8], simplifying that way the use of TensorFlow. Finally, in order to manage the score formats, Musescore [9] brought us the possibility to import and export the symphonies, so we could see the score and listen to it at the same time. It has many musical functionalities and it is an open source program available for every platform.

Data representation Several ways of representing Beethoven Symphonies scores have been studied for this paper.

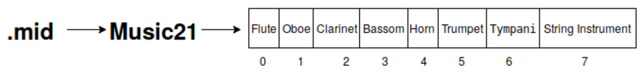

Figure 4: Music21 .mid files parsing

[7]http://web.mit.edu/music21/

[8]https://keras.io/

[9]https://musescore.com/

Firstly, as MIDI (.mid) files were popular in this research field, we used them as an input for our system, as it is a data file which contains information about the sounds: what note is played, when and how long or loud. As MIDI files store sound information, it doesn't differentiate between all the string instruments in an orchestra, since their pitches are really similar, as it can be seen in Figure 4.

Since the main goal of this paper is to obtain all the different scores for every orchestra instrument, the input files format were changed to MXL. This extension refers to a compressed music score, which Music21 easily processes. MXL files are the compressed format of the so called MusicXML, which is the standard XML format. Music21 allows us to generate the final output in any desired format, so we can obtain it in MIDI and XML. After getting those files, Musescore can open both formats so the score can be visualized and played.

In order to represent the output of the training, i.e. the weights of the different notes and durations, the model also returns an HDF5 file, *Hierarchical Data Format* version 5, commonly used to store big quantities of data.

Music Generation

In this paper, we have established two different approaches in order to obtain the expected result, which is the new Beethoven's Tenth Symphony. The first approach is based on generating all the different orchestra instruments scores individually. By training each instrument with a concrete existing set of symphonies, we have obtained each score. After that, we have manually joined all the different scores to study if the overall symphony was musically valid. Since each instrument was trained without information of the other instruments, the obtained conductor score had a lack of coordination between them.

The second approach was intended to increase the coordination between each instrument, so we have trained a set of instruments at the same time from a concrete set of symphonies. This way, the generated scores present a considerable increment of coordination and it is easier to differentiate each musical phrase.

Dataset description All the Beethoven symphonies have been converted to an mxl file, which constitutes the dataset that we have used to obtain the desired results. Also, the instrument or instruments with which the system works has to be established, so the Python module music21 can divide the mxl score into all the present instruments, and take only the choosed ones. Then, in the first approach, where the goal is to obtain each instrument's score individually, the note names and durations are stored in an independent file, being the different tuples of note names and durations the training data. The second approach trains with the chosen instruments at the same time, so we need to store, besides the note name and duration, the offset and instrument that plays it. The offset will be used to sort the data, but after making sure that the events are sorted as they are in the original score, it can be removed from the dataset. This way, the training data will be composed of the different tuples of note names, note durations and instrument.

At this point, we create a dictionary to convert from each data tuple to a number, so the neural network can work with it.

Training Finally, we can generate the network input and output data. By establishing a certain sequence length, the output for each input sequence will be the first note that comes after the notes sequence in the input. It is important to take into account that in case of establishing a big sequence length, the machine may generalize, while setting a small sequence length, the system may over learn.

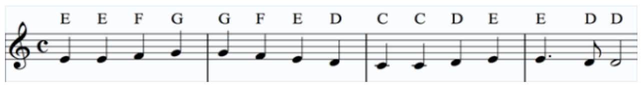

Figure 5: Violin's *Ode To Joy* snippet

For example, setting a sequence length equal to two, the first stages of the system's work flow for the Figure 5 input would be the shown in Table 1.

sequence_in	sequence_out
[(E, 1), (E, 1)]	[(F, 1)]
[(E, 1), (F, 1)]	[(G, 1)]
[(F, 1), (G, 1)]	[(G, 1)]
[(G, 1), (G, 1)]	[(F, 1)]

Table 1: Figure's 5 sequences

In case of the input, reshaping into a 3 dimension matrix is needed so it is compatible with the LSTM layers, using Python's numpy module. The first dimension or shape of the network is the number of different patterns obtained in the last step, the second one is the previously established sequence length and finally the last dimension is forced to be 1, so it has just one input information per sequence length. After that, the software normalizes the input into sequential values, from 0 to 1, to work with a regression model. In case of the output, it is converted into a categorical model.

The next step is to create the model, which follows a stacked LSTM architecture, since the larger the depth, the less neurons per layer the network needs, and it is faster (Graves, Mohamed, and Hinton 2013). There's no formula established to determine how many layers the network should have, and how many neurons would work better for each layer, so one of the tasks during the development of this project has been to obtain that information empirically.

The network is composed 3 different types of layers. The most relevant ones are the LSTM layers, which take the sequences and return new ones. Then, the Dropout layers prevent overfitting, ignoring randomly selected neurons during the training, setting those inputs to 0. The Dense (Density) layer serves as a full connection mechanism. This layer is the last one, so the system returns the same number of outputs as the different numbers of tuples (note name, note duration) the input data had. Finally, the activation function used for every layer is set, and it determines how each node's output is be represented. In this case, a linear activation is used, the softmax function, valid for multi-classification

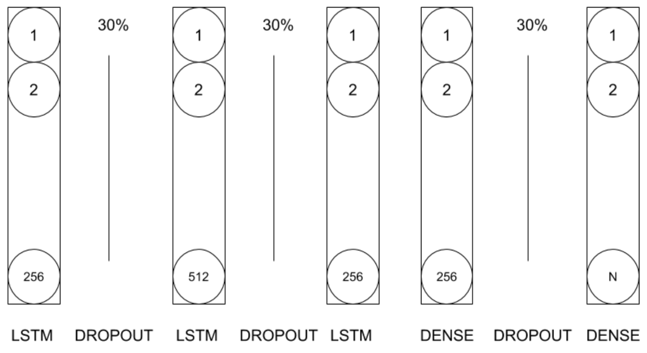

Figure 6: Model, being N the number of different tuples (note name, duration)

tasks, allowing the output to be interpreted as a probability between 0 and 1.

In this problem, since it is all about creativity, we do not have the final validation step on a not trained group of data, present in the majority of machine learning problems, due to the nonexistence of a correct solution.

Once the model is built and the input and output data are ready, it gets trained, generating an .hdf5 file containing the weights, or priorities, for the input notes.

Prediction For this task, the network input is generated again, as in the previous process (see Table 1). Since it needs to work over the same model, it is created again, with the same parameters, but now, instead of training the model, it loads the generated weights (hdf5 file) from the previous process. It is important at this point that the network input shapes and the loaded weights have the same dimensions. Once the model is ready, a matrix is created to convert from the network output to a tuple. If we are trying to generate a single instrument score, the tuple is composed of (note name, note duration), while if we want to generate a conductor's score composed of several instruments, the instrument identification will have to be included in the tuple. Then, a random sequence from the input is extracted and started to predict a fixed number of notes. As in the training, this random sequence has to be reshaped into a 3 dimension matrix. The first dimension corresponds to the number of sequences, which is always 1, the second to the length of the sequence and the third, as in the training, is forced to be 1. After that, all the sequence values are converted into sequential ones (between 0 and 1), so the model can return a prediction given those input values. The output of the prediction is an array with a probability for each tuple. Then, the system sorts the values from the greatest probability to the lowest. Once it has the indexes of the most interesting notes, the system can work on the given tuples accessing to the conversion matrix. It forces the predicted notes to have a duration greater or equal to 0.5 (quaver), for the score's simplicity. Another important restriction is to give priority to notes that belong to the key scale used in the new score, present in Figure 7. Some other restrictions manually made is that if the note with highest prediction differs more than one octave from the last one, it is transposed in order to get

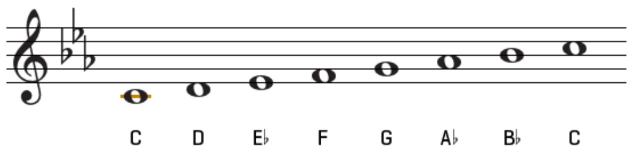

Figure 7: Key scale

closer to the previous one, but not modifying the note predicted, since it will not be easy to play for a musician. Another change made at this point is that if the predicted note and the previous one are rests, the lengths are added. This can only be applied to rests since we need to have several identical notes following (see Figure 1).

After choosing the most appropriate note, the index of the selected note is added to the pattern, which serves as an input for the next prediction.

Once the system has all the required predicted information (notes, chords, rests, and all the needed information such as their durations or the instrument that plays them) it is processed and .xml and .mid files are created using music21.

Results

The system output differs from the information given to the training, although once with the same trained data, the system predicts the same score, which denotes a lack of variability.

Approach 1: Generating individual instruments melodies Training with all the Fifth Symphony's movements, the output obtained is shown in Figure 8.

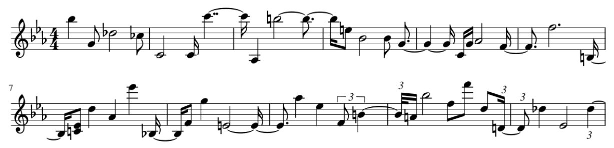

Figure 8: Results from training with the Fifth Symphony

At this point, the time measure is 4/4 as a first approach, although after discovering that Beethoven's house sketches belonging to the upcoming symphony had measure 6/8, it was set to that one.

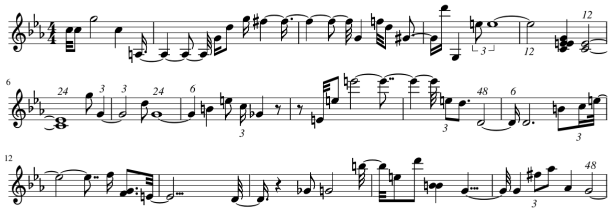

Figure 9: Results from training with the Fifth Symphony allowing rests

It can be seen that different measures showed up, such as quarter, eight, sixteenth or half notes but also thirty-second notes, and a motif shows up. In the first two staves, a half note appears tied to an eight and a sixteenth note in several compass. However, there are no rests, so the next step at this point was to retrain the system, again with the most famous symphony, but allowing rests to appear. The results can be seen in Figure 9. Again, although a different score is generated, we can distinguish some patterns in the composition.

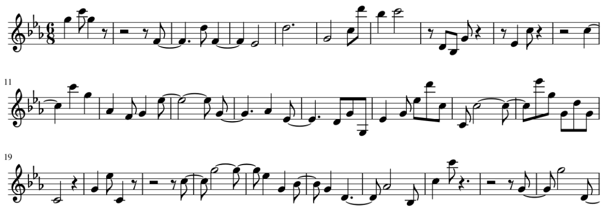

Figure 10: Results from training with the Fifth Symphony with manual improvements

At this point, the empirical restrictions explained above during the prediction are implemented, and all the experiments from this point include these manual improvements. For instance, using the same weights as before, the first three staves of the outcoming score is shown below in Figure 10.

The result differs from Figure 9, being the new one clearer but maintaining the motifs, such as the half note tied to two quarter notes, quality that characterizes Beethoven's Fifth Symphony.

Keeping the system state, we train it with the Seventh symphony, and generate the violins as before. The result looks similar, although it is remarkable the increment in the number of rests showing in the score. This may be due to the amount of silent compasses in the second movement of this symphony. Violins start playing in compass number 50, which is not a common characteristic of the violin scores in any symphony, being usually the instrument playing the main melody.

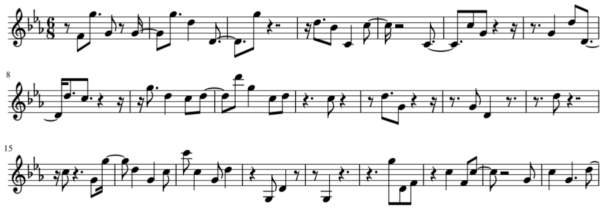

Figure 11: Results from training with the Seventh Symphony with manual improvements

Figure 11 shows only the first three staves, in which there's not an easy-to-recognize motive such as in the previous experiments. That may be because this symphony doesn't have a clear motive such as the Fifth's.

Now that we have concluded the experiment with the Fifth and Seventh symphonies, the next step is to train the system with both of them. The output is shown in Figure 12.

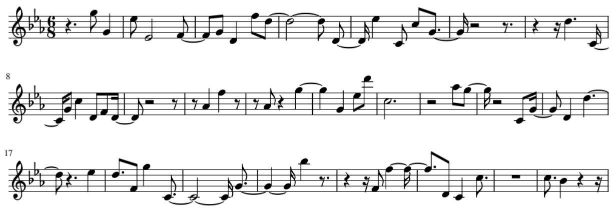

Figure 12: Results from training with the Fifth and Seventh Symphony

It can be seen that the amount of rest notes is increased from other results that doesn't use the Seventh symphony violin's as input, but the motives present in the output obtained from training with the Fifth keeps showing. The same happens in Figure 13, obtained from training the Fifth, Seventh and Ninth Symphonies Violins.

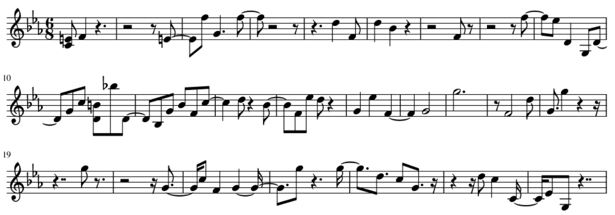

Figure 13: Results from training with the Fifth, Seventh and Ninth Symphony

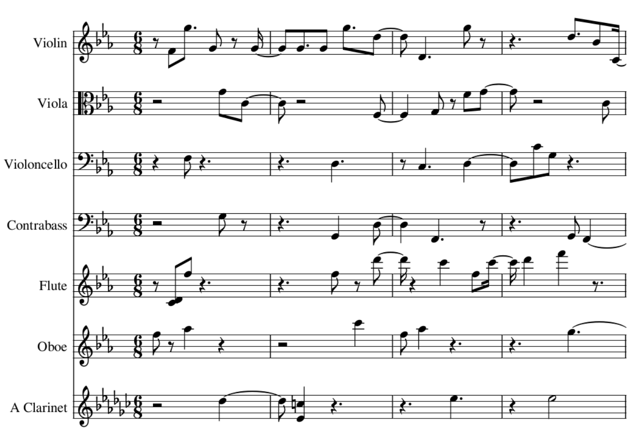

Figure 14: Results from training separately 7 different instruments with the Seventh symphony

Approach 1: Generating music for several instruments After completing all the experiments previously described, the system was trained with some of the orchestra's instruments. Figure 14 shows the prediction result for Violin, Violas, Violoncellos, Contrabass, Flutes, Oboes and Clarinets, training with the Seventh symphony. Although this result has been obtained from training each instrument individually and putting them together manually, it is distinguishable a lack of coordination between each instrument, since each melody has been generated without having knowledge on any other instrument's melody. That has caused that each musical phrase from the different instruments doesn't coordinate with the others to generate a group sound.

Approach 2: Generating several instruments at the same time To avoid the musical disorder obtained in the previous results, the second approach was used. As explained before, in this case the system is trained with a set of desired instruments, getting this way scores such as the one shown in Figure 15. This result shows how each instrument compliments the others, having the violin the main melody at the beginning, but respecting the Flute's main appearance in compasses seventh and eight.

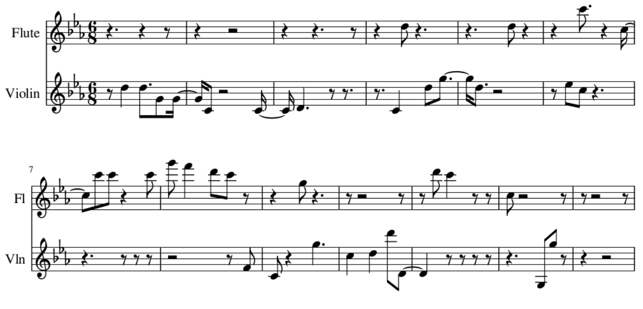

Figure 15: Second approach trained with the Seventh symphony for Flutes and Violins

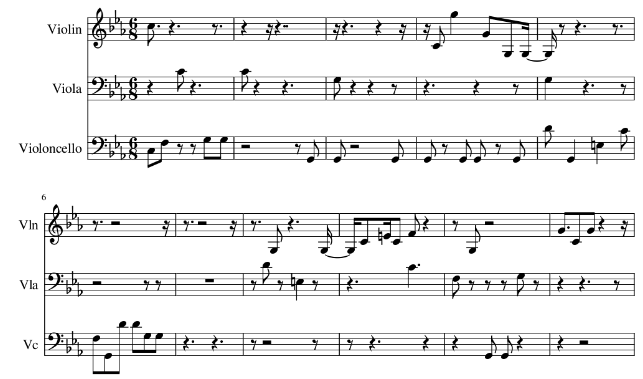

Figure 16: Score obtained from training Violins, Violas and Violoncellos with the Seventh symphony

The same behavior can be seen in the result shown in Figure 16, which shows how Violins, Violas and Violoncellos, while being trained only with the Seventh symphony, assumes a trio music by respecting the other instrument's melodies and complementing each other. It can be appreciated the differences between this score and the one shown in Figure 14. In that case, the corresponding lines are the first, second and third (Violin, Viola and Violoncello). As it can be seen, the coherence of the different instruments is enhanced in the second approach.

All the mp3 results are available in Github repository [10].

[10]`https://bit.ly/2tzuHBb`

Conclusions

This paper explores the possibility of generating new music based on the Beethoven's style by a system doted with Artificial Intelligence, using LSTM neural networks, which learn and remember musical phrases of a concrete length, finally showing that it is possible to obtain music that imitates this composer's style for several instruments.

During the specification of the problem, we established two ways of approximating to the new symphony. The first one was to train and generate separately each instrument scores, and manually creating the conductor's score. The results obtained were satisfactory for each single instrument separately, getting to generate music in a recognizable style. However, when joining all the different scores, the sound was not coordinated and the musical phrases belonging to the different instruments were not respected by the others. We concluded that with this first approach we could generate solo scores, but not group music. The second approach was intended to solve the main problem that the first one presented, that the instruments were not sufficiently coordinated since each instrument was trained separately, without any information on the music that the others were playing, which is crucial in an orchestra. The solution proposed was to train and generate music belonging to different instruments at the same time. This way the results obtained were more coordinated and we could see that each instrument respected each other, having rests or accompanying the main melody when they did not have the leading voice.

The amount of results obtained can be seen in Table 2, as we have progressively studied the output generated with both approaches, by first working on the generation of single instruments score, and checking that way if they were musically correct, to finally generate a conductor score. The system can return solo scores, but also duos, trios, quartets and an orchestra score, although we have not got to generate the score trained with all the existing symphonies.

The human interpreter is always the source of emotions, so it is remarkable the lack of dynamics in the generated music, being played all the notes at the same volume during the whole piece. In this paper we have focused in the notes production and instruments coordination, so generated scores have not notation of the dynamics.

Future work

Following the problem exposed in the conclusion, the next step is to research in music expressiveness, in order to transmit it to the system, to obtain music similar to what a human composer would create. An option to start in this task could be to obtain the score's dynamics, and train a Deep Learning model with the expressiveness of the work, in order to generate a *template*, which would be the equivalent to the composer's way to capturing his or her feelings. After generating the dynamics, the *"most human"* or sentimental part, a system like the one created for this work would generate the notes and they would be fitted in the dynamic's template. Another improvement that could be made to the developed system is to establish more elaborated musical rules to generate notes. For instance, taking First violin's melody as

Approach	Instruments	Symphony trained	Details	Figure
First approach	Violins	5th	Without rests	8
			With rests	9
			With rests and manual improvements	10
		7th	With rests and manual improvements	11
		5th + 7th	With rests and manual improvements	12
		5th + 7th + 9th	With rests and manual improvements	13
	Violins, Violas, Violoncellos, Contrabass, Flute, Oboe, A Clarinet	7th	With rests and manual improvements	14
Second approach	Violins, Flute	7th	With rests and manual improvements	15
	Violins, Violas, Violoncellos	7th	With rests and manual improvements	16

Table 2: Results

the main motive, while generating new instrument notes, it should be taken into account the harmony created between the notes, so a nice and clear sound is composed. For that purpose, some musical research about harmony effects and how it contributes to the perception of a musical phrase, (Palmer and Krumhansl 1987), should be considered.

The social awareness and unconcern should be progressively made, by calming down the latent discussion around Artificial Intelligence and the possibility of *stealing* human jobs. In case of this paper, the most affected community are the music composers, worried of being substituted by machines. This last fact should be contradicted by clarifying that Artificial Intelligence will work as a tool to enhance their production, but, at this point, it will not generate any score without a composer's help.

References

[Bharucha 1992] Bharucha, J. J. 1992. Musact: A connectionist model of musical harmony. In *Machine Models of Music*, 497–509. MIT Press.

[Biles 1994] Biles, J. 1994. Genjam: A genetic algorithm for generating jazz solos. In *International Computer Music Conference*.

[Cohen 1995] Cohen, H. 1995. The further exploits of aaron, painter. *Stanford Hum. Rev.* 4(2):141–158.

[Colton 2012] Colton, S. 2012. The painting fool: Stories from building an automated painter. In *Computers and creativity*. Springer. 3–38.

[Cooper 2000] Cooper, B. 2000. *Beethoven*. Oxford University Press, USA.

[Cope and Mayer 1996] Cope, D., and Mayer, M. J. 1996. *Experiments in musical intelligence*, volume 12. AR editions Madison.

[Ebcioglu 1990] Ebcioglu, K. 1990. An expert system for harmonizing chorales in the style of j.s. bach. *The Journal of Logic Programming* 8(1):145 – 185. Special Issue: Logic Programming Applications.

[Gardner 2016] Gardner, L. 2016. Beyond the fence review – computer-created show is sweetly bland. *The Guardian*.

[Gervás et al. 2005] Gervás, P.; Díaz-Agudo, B.; Peinado, F.; and Hervás, R. 2005. Story plot generation based on cbr. *Knowledge-Based Systems* 18(4):235 – 242. AI-2004, Cambridge, England, 13th-15th December 2004.

[Graves, Mohamed, and Hinton 2013] Graves, A.; Mohamed, A.; and Hinton, G. E. 2013. Speech recognition with deep recurrent neural networks. *CoRR* abs/1303.5778.

[Henry and Massin 2006] Henry, J., and Massin, B. 2006. *Mozart the freemason: the masonic influence on his musical genius*. Inner Traditions.

[Hiller and Isaacson 1958] Hiller, Jr., L. A., and Isaacson, L. M. 1958. Musical composition with a high-speed digital computer. *J. Audio Eng. Soc* 6(3):154–160.

[Kalingeri and Grandhe 2016] Kalingeri, V., and Grandhe, S. 2016. Music generation with deep learning. *CoRR* abs/1612.04928.

[Liang et al. 2017] Liang, F. T.; Gotham, M.; Johnson, M.; and Shotton, J. 2017. Automatic stylistic composition of bach chorales with deep lstm. In *ISMIR*, 449–456.

[Mantilla 2019] Mantilla, J. R. 2019. Um algoritmo completa a misteriosa 'sinfonia inacabada' de schubert. *El País*.

[Montfort et al. 2012] Montfort, N.; Baudoin, P.; Bell, J.; Bogost, I.; Douglass, J.; Marino, M. C.; Mateas, M.; Reas, C.; Sample, M.; and Vawter, N. 2012. *10 PRINT CHR (205.5+ RND (1));: GOTO 10.* mit Press.

[Nayebi and Vitelli 2015] Nayebi, A., and Vitelli, M. 2015. Gruv: algorithmic music generation using recurrent neural networks. *Course CS224D: Deep Learning for Natural Language Processing (Stanford)*.

[Pachet 2003] Pachet, F. 2003. The continuator: Musical interaction with style. *Journal of New Music Research* 32(3):333–341.

[Palmer and Krumhansl 1987] Palmer, C., and Krumhansl, C. L. 1987. Pitch and temporal contributions to musical phrase perception: Effects of harmony, performance timing, and familiarity. *Perception & Psychophysics* 41(6):505–518.

[Pestelli 1984] Pestelli, G. 1984. *The age of Mozart and Beethoven*. Cambridge University Press.

[Quintana et al. 2013] Quintana, C. S.; Arcas, F. M.; Molina, D. A.; Fernández, J. D.; and Vico, F. J. 2013. Melomics: A case-study of ai in spain. *AI Magazine* 34(3):99–103.

[Ritchie 2009] Ritchie, G. 2009. Can computers create humor? *AI Magazine* 30(3):71.