Homophonic Music Composition Using Pipelined LSTMs for Melody and Harmony Generation

Clément Saint-Marc
Graduate School of Computer and Information Sciences, Hosei University
clement.saint-marc.2d@stu.hosei.ac.jp

Abstract

Throughout the years, many attempts have been made at creating music procedurally. However, very few of those attempts were concerned with the actual "meaning" of the music that was generated. The goal of this research is to implement a pipelined model of neural networks capable of generating homophonic music without input. The generated music should be "meaningful", that is to say, it should sound like it has a purpose. This is similar to how a human composer would write music. This idea of "meaning" makes a composition tell a story or express feelings. This is the reason humans write music, as well as create other forms of arts. As the goal of artificial creativity is to approach human creativity as close as possible, Artificial Intelligence should try to imitate humans as close as possible. Therefore, it is important for a music generating AI to understand "meaning" in music. In order to introduce meaning in AI composition, the process is approached step-by-step. Dedicated neural models trained on melodies that express purpose through the use of motifs are first used to generate a melody. That melody serves as input to another neural model, trained on chord progressions that contextualize the melodies they accompany, to generate harmony. Besides the model being symbolic, there are no musical constraints or guidelines for generation. By using this pipelined but unconstrained approach, it is possible to procedurally generate music that sounds as if composed by a human being.

1 Introduction

To define "meaning" in music, it is first necessary to identify its key components. In modern western music, those components are melody, harmony, and rhythm [1].

A melody is a linear succession of notes, that can be seen as a combination of pitch and rhythm. It usually consists of motifs, which are small musical phrases [2], used to convey an idea. Melody is generally seen as the "horizontal" aspect of music [3]: the relationship between notes across time. Conversely, harmony is seen as the "vertical" aspect of music [3]: it is the relationship between multiple notes at the same time. This is usually represented in music as chords [4]. Harmony is used in composition to give a melody context, in the form of a chord progression [5].

Rhythm in music is defined as the timing of events on a human scale. More specifically, this means rhythm governs the timing of notes in a musical composition. This is usually represented as two things. The first one is a tempo, which is the number of beats per minute (a beat being the basic unit of time in music [6]), and the second one is a time signature, which is the number of beats per measure. From these definitions, we can already deduce that rhythm is implied by melody, as the latter is strongly related to time. Therefore, rhythm as a musical component is not what gives a musical piece its "meaning". Furthermore, we can also deduce that harmony exists to support a melody, as it gives it context. Thus, it appears as evident that the most important musical component in giving a musical piece its identity and its "meaning" is the melody, as the other components revolve around it.

Therefore, it becomes necessary to understand what makes a melody. As we have already defined, a melody consists of motifs, which are short successions of notes. Those motifs can have variations, which for the purpose of this research can be defined as small changes to those motifs (either in pitch or in rhythm). By using motifs and variations, a melody can be given purpose. It can tell a story, and is therefore given meaning.

In order to approach composition procedurally, it is necessary to separate the composition process into several sub-processes. For the purpose of this research, this can be summarized into two elements: melody generation and harmony generation. Melody generation should be done using a seed obtained by randomly sampling from a distribution, which means that no initial input is required. It should also be set to a fixed number of measures, so that it can be repeated, and used for harmony generation. The harmony generation process should generate a chord progression that effectively supports a generated melody, using that melody as input data.

This approach to the composition process is rather standard and is the simplest way of composing music [7]. Therefore, music composed in such a manner would

Supervisor: Prof. Katunobu Itou

be akin to that of a human composer, while the process of composition itself remains simple.

It now becomes necessary to determine what kind of music should be used as training data for the Artificial Intelligence. Since the most important feature is meaning, it should be music that uses motifs in the most effective way possible. Such use of motifs is usually found in video game music, or movie scores. This is due to the necessity in such genres to represent ideas, characters, or locations. By using motifs to compose a meaningful melody, a composer can represent the story of a video game or a movie as music. It is also necessary to have harmony (in the form of a chord progression) that supports the melody, to give it context, which helps in telling that story.

Therefore, the training data should consist of such genres of music, and have both melody and harmony. Furthermore, it should be possible to separate the melody and the harmony to train two separate AI models, one specializing in melody generation, and the other in harmony generation. In addition, the music data should be high-level symbolic representations, since a discrete representation of music data is necessary for the model to learn the underlying relationships between notes. To this end, the MIDI format is used for input and output in this research.

Three datasets have been used in this research so far: The Nottingham dataset, consisting of around 1000 folk songs in MIDI format, with one track for melody and one for harmony. This dataset is not video game music or movie score, but it was still useful in studying relationships between melody and harmony.
The second dataset, which was used to train the models for eventual data generation, consists of only around 20 manually transcribed pieces (with melody and harmony), mostly from video game soundtrack.
The third dataset, which is currently being used, is the TheoryTab Database (video game), consisting of around 2000 tunes from video game soundtrack in MIDI format, with melody and harmony on separate tracks.

2 Existing research

The existing research in music generation by AI has so far only consisted, when it was not mere single-line melody generation, in approaching the overall music composition problem in a polyphonic manner [8, 9], without any real attempt at focusing on any sense of purpose in the generated music. In our research, a homophonic approach is preferred, where the melody is generated first.

Most previous research, in the case of homophonic music generation, used chord progressions as input for melody generation. In that research, the chord progression is either pre-written by humans [10, 11], or has to be generated by the model [13].
Generating a chord progression is merely a complication of the task of melody generation, and in the previously mentioned research, the training and generation processes are in fact constrained by a structure of music theory, such as the 12 bar blues format.
Our research uses a text generation approach for melody generation. Although such an approach has been used before, it has been done using arbitrary notation systems, such as the ABC system, as mentioned in [14], where the text data does not *directly* represent the music. The melody generation approach in our research focuses on the horizontal aspect of melody in music by using individual characters as units of note duration.

As previously explained, most research where generating harmony is necessary as a process do so as a sort of melody generation: the chords are generated over time, and determine the structure of the generated music. In our research, harmony is considered a secondary aspect of music: it serves only the purpose of giving a context to a melody. As such, the idea of chord "progression" is never considered, and the chords for a given melody are generated independently from one another, depending solely on the melody.

Finally, in most of the previous research, the chosen method of data representation is that of absolute pitches. This is due mostly to the ability to perform data augmentation if necessary (by simply shifting the key of every individual piece to all 12 notes of the chromatic scale). Although a delta relative approach of music data representation has been used before in this field [14], where the current note is expressed as its difference in semitones with the previous one, that relative approach still does not make implicit patterns such as motifs and variations explicit enough for the model to learn them.
In our research, we use intervals relative to the key for data representation. Such relative intervals completely remove the context of key in the data, meaning the pieces are all uniformized. Not only does this naturally provide more data for implicit patterns, but it also makes them more explicit themselves in the data.

3 Method

3.1 Representation of music data

To represent music data, intervals relative to the key of the musical piece were chosen, instead of using absolute pitch representation. This is akin to *movable do solfège*, where each degree in the musical scale is expressed as a different syllable (i.e. *Do, Di, Ra, Re,* etc.).

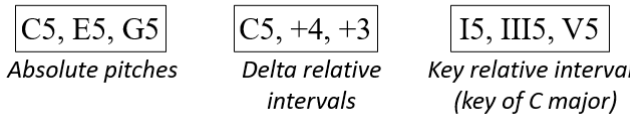

Figure 1. Kinds of music data representation.

The main reason for choosing this representation is that it makes the key of the musical piece irrelevant — only the relationship between the different notes appears. This in turn makes simpler patterns like intervals appear explicitly in the training data, making the learning of more complicated patterns, such as motifs, easier. This is also preferable to a purely “delta relative” representation, where each note is represented as the number of semitones relative to the previous note. While this method also makes key irrelevant, it has one major issue if used for melody generation: when generating a melody, should the model make a mistake and predict the wrong interval, the rest of the generated music would be in the shifted key [14].

The problem of using intervals relative to the key, however, is that the key of the piece should be explicitly given in the data, or determined algorithmically.While there are many algorithms that attempt to determine the key signature of a song [15], none of them are entirely accurate in doing so. Furthermore, some pieces may have several key changes at arbitrary points, making the process even more difficult. The Nottingham dataset, which was previously mentioned, explicitly indicates key signatures in its data. The key signatures were also encoded in the manually transcribed dataset. For the TheoryTab dataset, as there are almost no key changes in the individual MIDI files due to them being only sections of a tune, a key finding algorithm is used (the Krumhansl-Schmuckler algorithm). As it gives a list of probabilities for each possible key, an “interpretation” of a given section is kept for each key where that probability is above a given threshold. This also provides a form of data augmentation. The probability is used as a weight during training.

The melody data is processed by the neural models as text, with a single character representing one unit of time. This allows for direct representation of note duration within the time dimension of the model. The first unicode character outside the ASCII range, “Ā”, is used to represent one unit of time.

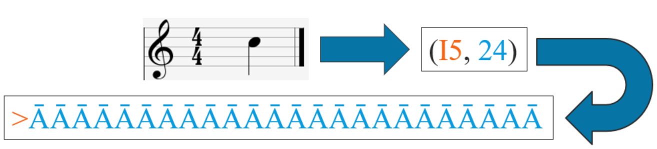

Figure 2. Example of a note represented as text.

Choosing such a short triplet as a unit allows for even division of other bigger binary and ternary durations.
As for chords, they are represented using their root, relative to the key, along with their quality. To this end, a simple chord naming algorithm has been implemented, which, similarly to what the key finding algorithm does, gives a list of possible chord names for each set of notes, along with an associated probability, used as a weight during training.

3.2 The Neural models

The kind of neural model chosen for implementation is a Recurrent Neural Network (RNN). This is because RNNs have great flexibility in learning complex patterns in musical data, thanks to their effectiveness at modeling temporal patterns [12]. This effectiveness with temporal sequences is due to their use of feedback loops and/or memory units to retain information over time during training. Even more specifically, RNN architectures such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) are especially well suited to this problem, as they can learn complex patterns across several time steps [11, 12, 13].

For the actual melody and harmony generating models implemented in this research, bidirectional LSTMs are used, as they are able to access a more global context [16] when training, which is necessary when composing music since a given note depends on the notes both preceding and following it. In order to implement the step-by-step nature of the composition approach, the neural models are arranged in a pipeline: the output of a given model serves as input for the following one.

3.3 Melody generation

The process of melody generation is separated into two smaller processes. The first one is *seed* generation, where an LSTM trained on melodies is used to generate a seed by sampling data randomly from its latent distribution. The second one is the actual *melody* generation, using a bidirectional LSTM trained on the same melodies to generate a melody from the previously generated seed. Due to the small amount of training data available with the second dataset, an algorithm that extracts the outline of melodies has been implemented. It is used to extract the outlines of the melodies used for training the melody generator. The extracted outlines are used as additional training data.

The training data consists of melodies in MIDI format. The melody notes are converted into intervals, with each interval (one for each note) being a tuple consisting of the relative scale degree (including pitch height) and the time relative to the previous note. Each interval is then converted into an ASCII character based on its value. This is done for each “interpretation” of the melody. Each melody is thus converted and added to the training set, with a special character, “ā” (the second unicode character outside the ASCII range), being used as a delimiter between the melodies.

The melodies are split into sequences of 50 characters overlapping each other, and converted into one-hot vectors. A greater overlap is used for the seed generator data (to provide more sequences), as the outlines are not used for training in this case. The network is then trained on those vectors for 15 epochs.
The following figure shows its topology.

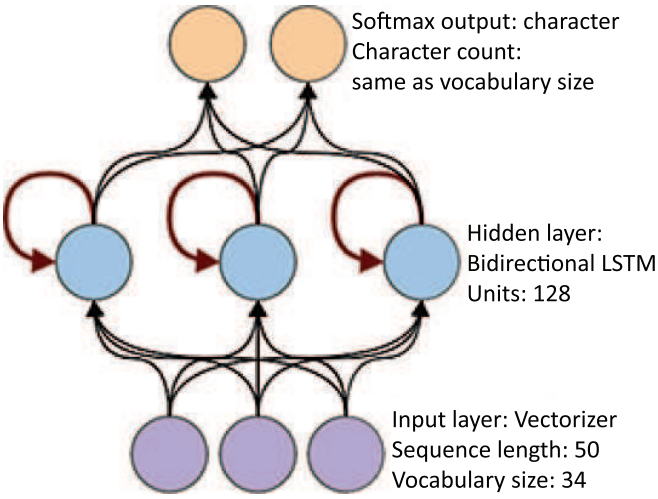

Figure 3. Topology of the melody generator.

The topology of the seed generator is the same save for the hidden layer, using a simple LSTM layer instead. This seed generator is trained for 10 epochs. The output is a prediction of the next character. The activation function for the output layer is softmax.
The prediction process is repeated iteratively for the desired length of the melody to be generated. In the case of the seed generator, it is sampling from the latent distribution of the data that is repeated iteratively. Once the melody generation process is done, the predicted text is converted back into intervals, then back into MIDI format.

3.4 Harmony generation

To study the relationship between melody and harmony, which is necessary to implement the harmony generator, a simple GRU-based classifier was implemented to label chord symbols to snippets of melodies from the Nottingham dataset.

The training data consists of melody snippets labeled with chord symbols. Each melody snippet consists of notes represented as words, themselves consisting of the time relative to the previous note (relative to the chord in the case of the first note of the sequence), and the note itself, represented as a relative scale degree, including its pitch height.

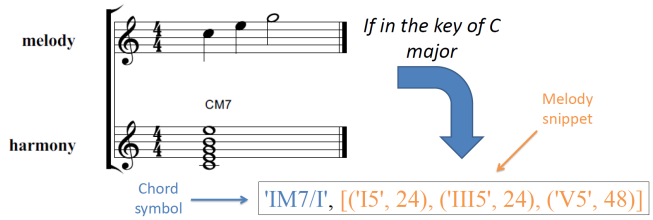

Figure 4. Example of a melody snippet over a chord represented as a sequence.

The input data consists of those sequences of notes, vectorized as sequences of a maximum length of 50 words. The vocabulary size of the vectorizer is 250 words. This input is processed by a GRU layer consisting of 128 nodes. In this case, a GRU layer was used due to getting better results than with an LSTM layer. The output is a prediction of the chord symbol corresponding to the input melody snippet. The activation function for the output layer is *softmax*.

Once the relationship between melody and harmony had been studied, and promising results obtained, the harmony generator could be implemented.
The following figure shows the topology of the harmony generating RNN.

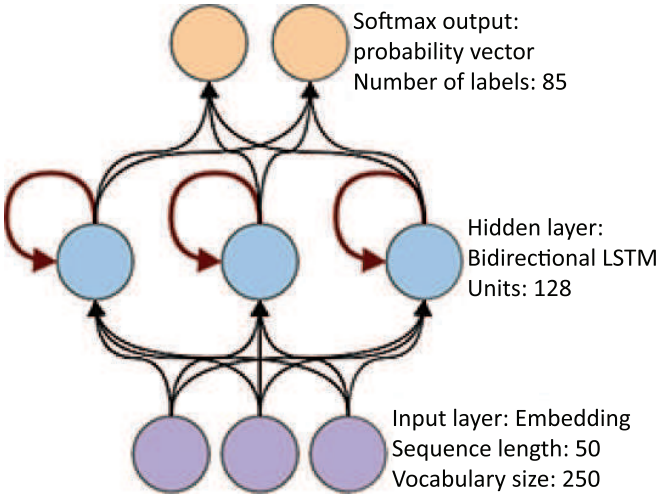

Figure 5. Topology of the harmony generator.

As there are more chord labels than with the previously used dataset, a bidirectional LSTM layer with 128 units is used this time. The principle is otherwise exactly the same.

During generation, given a generated melody as input, a pre-processing algorithm divides that melody into snippets of around the same length: four quarter notes. Those snippets are then used as the inputs for the harmony generator, and a chord is predicted for each one.

4 Evaluation of output

4.1 Overview of the training

Unless otherwise specified, the following training results were obtained with the manually transcribed dataset of around 20 pieces.

After training for 10 epochs on 74580 samples, the seed generator reaches a loss of around 0.2. As for the melody generator, it reaches a loss of around 0.12 after training for 15 epochs on 98583 samples.

For the GRU-based classifier trained on the Nottingham dataset, a validation accuracy of around 78% was reached for a training accuracy of around 83% after training for 50 epochs.

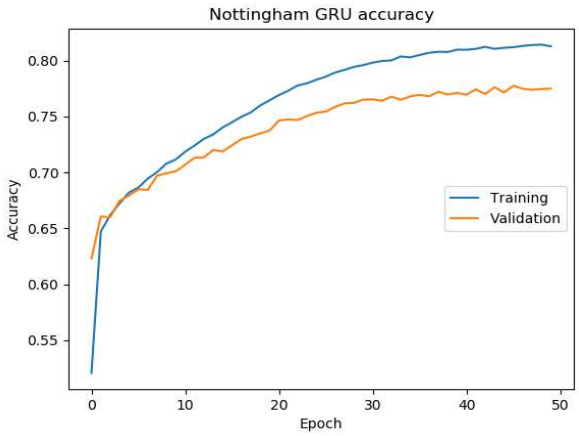

Figure 6. Accuracy curve for the Nottingham classifier.

Although this suggests a slight overfitting, 78% is still more than acceptable for predicting something as arbitrary as a chord based on the melody it corresponds to. Those results show promise for the harmony generator. However, unlike the GRU-based chord classifier, the harmony generator (bidirectional LSTM) does not converge during training (the accuracy remains at around 30%).

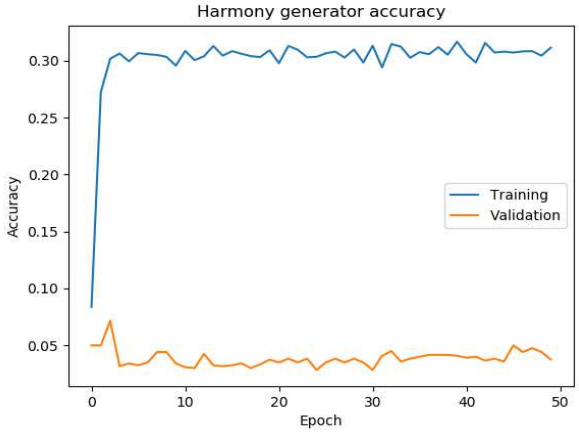

Figure 7. Accuracy curve for the harmony generator.

However, this is most likely due to there being a lot of chord that are similar in feeling, but have a different chord name. Chords such as C and CMaj7 are virtually the same since the latter contains all the notes of the former. Despite this issue, good results are still obtained during generation. It will be addressed in the future in the form of latent semantic analysis of the chord labels.

4.2 Subjective evaluation

All the pieces used for subjective evaluation were generated after training with the small manually transcribed dataset. The currently implemented model has no notion of tempo, and generates pieces of a fixed length (16 measures, assuming a tempo of 120bpm and a time signature of 4/4).

The melody generation, despite not always being consistent, still shows good results with interesting melodic movement, use of motif and variation, as well as melodic ornaments. Currently, the main issue of melody generation is a lack of a sense of overall structure. Overall structure in a melody is necessary for it to sound natural, and serve an overall purpose.

The harmony generation is very satisfactory given the current poor training results. Natural-sounding chord progressions and cadences are obtained, despite the model not knowing the previous chords when generating. One issue of harmony generation lies in the rhythmic consistency. As a time signature of 4/4 is assumed when cutting a melody into snippets, the rhythm of the chord progression gets gradually shifted relative to the melody. This leads to some awkward pauses in the progression.

To conduct proper evaluation of generation results, a survey using excerpts (of around 12 seconds) of 5 AI generated and 5 human-written pieces has been set up. The 10 excerpts are arranged in 5 pairs; 1 excerpt being from an AI generated piece and the other from a human-composed piece. For each pair, the respondent has to guess which one was generated by AI, and explain their choice. Overall, with 8 respondents, some of whom were musically trained, correct guesses amount to 50% of the answers.

Some of the respondents who were able to guess correctly said the melody sounded unnatural, or as though the notes were randomly extracted from a scale. There were, however, respondents who were mislead, and had justified their choice by saying that the human-written piece had melody notes that were dissonant with the harmony, or that the melody was using the same notes in a row.
Some also mentioned regularity in the melody of the AI generated music as a deciding factor for choosing the human-written piece. Most of the respondents who were able to guess correctly mentioned the rhythm feeling unnatural in one way or another: either there were awkward pauses in the chord progression, or the melody seemed "out of rhythm".

Despite the rhythm being the most evidential way to discern the human-composed music from that generated by the model, as there is no meter constraint for generated melodies, it seems people who are not musically trained would be less likely to notice this, and therefore be more likely to be mislead.

5 Conclusion

The objective of this research was to implement a machine learning model capable of generating meaningful-sounding music without any initial input.

In this paper, a new approach for artificial music composition is proposed. This approach is homophonic, and is centered around the generation of a single-line melody to which chords are found

individually. By focusing on the temporal aspect of melody via character-level representation, then contextualizing melodies through the use of chords, and by using a key-irrelevant method of note representation, it is possible to generate music that sounds meaningful at times, as generated melodies make natural use of motifs and variations, although there is still a certain lack of overall purpose.

The melody generating model, despite not being consistent overall in that regard, was also able to learn a certain rhythmic regularity, which given the lack of fixed meter in the generated music indicates the efficacy of the character-level text representation.
The harmony generating model shows very interesting results, as the chords are almost always consonant with the melody, and despite them being predicted individually, a sense of movement is still obtained.

6 Future work

Work is currently being done to set up the TheoryTab dataset for use with the model. The topology of the networks might have to be complexified, as this dataset has more than twice as many characters and chord labels as the one used previously. This dataset also provides metrics such as melodic complexity or chord-melody tension for every piece. Those metrics will be used to separate the dataset into several subsets as an experiment to determine what factors would be more desirable for training.

One of the most important next steps will be the implementation of the melody generator as a hierarchical LSTM. Such a model would be able to learn the structure of a melody, such as $note \rightarrow motif \rightarrow phrase$. This will in turn provide more consistent use of motifs, as well as help provide a sense of overall structure in melody generation, thereby making generated melodies sound more natural and purposeful.

Another important next step is the implementation of chord embeddings, generated from latent semantic analysis of the chord labels. This will greatly help improve the training of the harmony generator, as well as generation results.

Another improvement to harmony generation would be a better snippet cutting algorithm. An algorithm that detects motifs or phrases within a melody and cuts it accordingly would provide a more natural rhythm to chord progressions than simply cutting every measure of 4/4.

Finally, tempo will also have to implemented as a feature. This will most likely involve an additional neural model that would predict a corresponding tempo for a given melody.

References

[1] Thomson, Virgil (1957). "Introduction" to Robert Erickson. The Structure of Music: A Listener's Guide: A Study of Music in Terms of Melody and Counterpoint. New York: Noonday Press.

[2] New Grove (1980). cited in Nattiez, Jean-Jacques (1990). Music and Discourse: Toward a Semiology of Music (Musicologie générale et sémiologue, 1987). Translated by Carolyn Abbate. Princeton, NJ: Princeton University Press. ISBN 0691091366/ISBN 0691027145.

[3] Jamini, Deborah (2005). "Harmony and Composition: Basics to Intermediate", p. 147. ISBN 1-4120-3333-0.

[4] Malm, William P. (1996). "Music Cultures of the Pacific, the Near East, and Asia", p. 15. ISBN 0-13-182387-6.

[5] Dahlhaus, Car. "Harmony". In Deane L. Root (ed.). Grove Music Online. Oxford Music Online. Oxford University Press.

[6] Berry, Wallace (1976/1986). "Structural Functions in Music", p. 349. ISBN 0-486-25384-8.

[7] Translation from Allen Forte, "Tonal Harmony in Concept and Practice", third edition (New York: Holt, Rinehart and Winston, 1979), p.1. ISBN 0-03-020756-8.

[8] G. Brunner, Y. Wang, R. Wattenhofer and J. Wiesendanger, "JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), 2017

[9] N. Boulanger-Lewandowski, Y. Bengio, P. Vincent, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription", 2012 International Conference on Machine Learning (ICML 2012), 2012

[10] M. Kikuchi, Y. Osana, "Automatic melody generation considering chord progression by genetic algorithm," 2014 Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC 2014), 2014

[11] Johnson, Daniel, Keller, Robert, Weintraut, Nicholas. (2017). "Learning to Create Jazz Melodies Using a Product of Experts"

[12] Johnson, Daniel D. (2017) "Generating Polyphonic Music Using Tied Parallel Networks."

[13] Douglas Eck and Juergen Schmidhuber. 2002. "A First Look at Music Composition Using LSTM Recurrent Neural Networks". Technical Report. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale.

[14] Briot, Jean-Pierre, Gaëtan Hadjeres and François Pachet (2017). "Deep Learning Techniques for Music Generation - A Survey." ArXiv abs/1709.01620

[15] Levine, Nathan J. (2015), "Exploring Algorithmic Musical Key Recognition". CMC Senior Theses. Paper 1101.

[16] T. Jiang, Q. Xiao and X. Yin, "Music Generation Using Bidirectional Recurrent Network," 2019 IEEE 2nd International Conference on Electronics Technology (ICET)