Modeling Polyphony with Neural Networks

Francisco Elías Montañez
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
monta215@morris.umn.edu

ABSTRACT

JamBot is a framework capable of modeling polyphonic music by learning meaningful representations of chord embeddings and using that as guidance for generating pleasing polyphonic music. It utilizes two long short-term memory networks, the first for generating a chord progression which is used for structural guidance by the second network for its generation of polyphonic music.

Keywords

Machine Learning, Neural Networks, Polyphonic Music

1. INTRODUCTION

Musical metacreation is a fascinating sub-field of computational creativity that focuses on providing machines with the ability to achieve musical tasks. The idea of algorithmic music composition has been around for years, stretching as far back as the 1700s where a German system called Musikalisches Würfelspiel used dice to randomly generate music from a series of options [7]. With the rise of computers, advancements in the field of artificial intelligence and increased computational power, algorithmic music composition has advanced rather quickly. Tasks have gone from generating simple melodies and accompaniments to full scale compositions.

In this paper we analyze JamBot [2], a framework created by students at the Swiss Federal Institute of Technology in Zurich, that is capable of modeling polyphonic music by learning meaningful representations of chord embeddings and using that as structural guidance for generating pleasing polyphonic music. We begin by introducing basic background information of both music and machine learning in Section 2. We follow by presenting the JamBot [2] framework, analyzing its structure and process for generating polyphony in Section 3. Then we present our results in Section 4 followed by a conclusion in Section 5.

2. BACKGROUND

In this section, we explain concepts and terminology needed to understand the rest of the paper.

2.1 Machine Learning

UMM CSci Senior Seminar Conference, December 2018 Morris, MN.

Machine learning is a sub-field of artificial intelligence in which statistical techniques are used to provide computer systems the ability to learn from data and make predictions or determinations from what it has learned.

2.1.1 Neural Networks

Artificial neural networks are frameworks inspired by the human brain that are designed to recognize patterns in data and perform tasks generally without being programmed. They are composed of interconnected units referred to as nodes or artificial neurons. These nodes form separate layers known as the input layer, hidden layer(s), and output layer. The edges connecting nodes between layers have weights that assign significance to nodes. These weights can amplify the nodes, making them more influential in the following layer or dampen the nodes, making them less influential in the following layer.

Input nodes provide external information to the network. No computations are performed here. The information is simply passed on to the nodes in the hidden layer. When the information moves from the input to hidden nodes, each input is multiplied by its respective weight. Those results are summed up and the sums move to the hidden nodes where they are run through an activation function. An activation function is a non-linear function used to introduce non-linearity into its result. By introducing non-linearity into the network, the network is able to learn and model more complex kinds of data such as audio or images. The activation function takes one number and applies a fixed mathematical operation on it. The result is then sent to the output nodes where similar computations are performed. The output nodes give the final result, or prediction of the network. In a neural network inputs are given, activation functions do not change, but weights do change. We go over this in more detail in Section 2.1.2.

A feed-forward neural network is a network in which information moves in one direction, from input layer to hidden layer to output layer. Feed-forward networks have no notion of time since they only consider current input. This limitation affects their performance with time dependent tasks or sequences.

A recurrent neural network is a neural network in which connections between nodes can make a cycle. A recurrent node stores previous input and merges it with its current input. This gives the network a sense of memory about previous information. The ability to consider previous input makes recurrent neural networks more efficient with time dependent tasks or sequences than feed-forward networks. Theoretically, recurrent neural networks have infinite mem-

ory. In practice however, they are limited to looking back a couple of steps.

2.1.2 Training

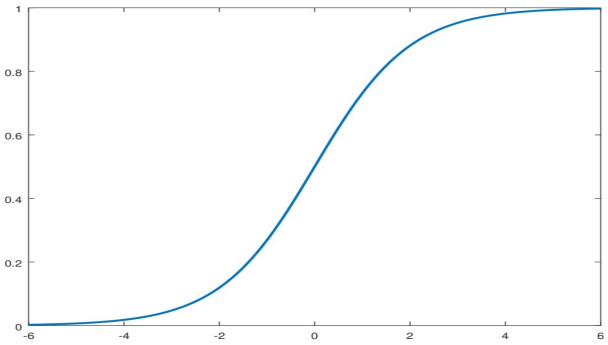
Training is the process of steadily improving machine learning algorithms ability to make predictions. Training begins with the dataset being split into two sets, the training set and the testing set. In supervised training, each dataset sample has an expected output called a label. Supervised training works by feeding an algorithm the labeled training set, comparing the network's outputs to the labels and adjusting the weights throughout the network in order to improve the accuracy of the network.

For this to work, a loss function is used to determine how close output values are to labels. A loss function with a lower value signifies higher accuracy while a loss function with a higher value signifies lower accuracy. The purpose of training is to minimize the loss function thus reducing the difference between predicted values and labels. In order to minimize the loss function gradients must be found. A gradient is the direction and magnitude of a loss function. Gradients are calculated via chain rule from Calculus in backpropagation. Backpropagation is a supervised learning algorithm used to calculate gradients with respect to the weights in the network. The calculated gradients are used to update the weights in the network accordingly via gradient descent, an optimization method that updates weights starting from the output layer and moves backwards towards the input layer. This process is repeated until the network's predicted outputs are reasonably close to the labels. At this point we can say the network has been trained. From here, the testing set is run through the network to evaluate its performance. We want to have low value loss functions as that indicates the network is able to generalize on new unseen data. If the values of the loss functions are high, we can determine that the network is unable to generalize on new data. This is known as *overfitting* which we discuss more in detail in Section 2.1.3. In this context, the term "generalize" refers to how well the patterns learned by the network apply to examples not seen by the network during training.

2.1.3 Training Difficulties

Difficulties may arise during training. Among the most common difficulties are vanishing gradients, exploding gradients, and overfitting.

Vanishing gradients are present with gradient-based learning methods and backpropagation. This problem is caused by the activation function chosen. Activation functions such as the commonly used sigmoid function compress their input into the range of 0 and 1.



Backpropagation calculates gradients by using the chain rule in Calculus. When calculating the gradients of earlier layers, the gradients which were compressed by the activation function, in this example the sigmoid function, go through series of multiplications. This causes the gradients to begin to decrease in magnitude exponentially [5]. When the gradients become too small, weights are updated by those extremely small values causing a slowdown in training and in some cases, preventing the network from learning.

Exploding gradients are the opposite of vanishing gradients. They occur when gradients that are greater than 1 are propagated back through the network layers. As they are propagated backwards, gradients accumulate through exponential growth caused by the series of multiplications [5]. This "explosion" of gradients creates an unstable network that is unable to learn from a training set.

Overfitting is a term used to describe a network that models the training set too well causing it to be unable to generalize on new data. In predictive modeling, the true underlying pattern we wish to learn is referred to as the "signal" while irrelevant information is referred to as "noise". Overfitting happens when a network learns or "memorizes" the noise of the training set. This can happen by not having sufficient samples in the training set for the network to learn the actual underlying pattern. We say a network "overfits" when it performs well with the training set but poorly with the testing set.

2.2 Musical Concepts

2.2.1 Terminology

A note is the combination of pitch and duration of a sound. In a typical piano keyboard, there are seven white-colored notes represented by the following letters of the English alphabet: A, B, C, D, E, F, and G. There are five black-colored notes, referred to as accidentals, which represent C♯/D♭, D♯/E♭, F♯/G♭, G♯/A♭, and A♯/B♭. This twelve note pattern is repeated throughout the length of the keyboard. The start of a repetition is called an octave. An octave is the distance between two notes with the same letter representation.

A chord is a combination of two or more notes played together. A series of chords is known as a chord progression. Chord progressions are usually accompaniments to melodies. A melody is a sequence of notes that tend to be the main theme of a song. Playing melodies along with chords or other accompaniments produces harmony, a pleasing effect created by the interaction of notes being played simultaneously.

For the pleasing effect to occur, melodies and accompaniments must be in the same scale. A scale is a sequence of small intervals of notes from which melodies and harmonies can be created. The most common scales are the major, natural minor, harmonic minor, melodic minor and the blues scale. The first note of a scale is called the root note. The combination of a scale and a root note define what is known as a key. To keep track of time, music is usually divided into measures. Measures are sections of equal time length consisting of equal number of beats.

2.2.2 Monophonic vs Polyphonic

Texture is a basic element of music that describes musical layers in terms of number and purpose. For the purposes of this paper, we will focus on monophonic and polyphonic textures.

Figure 1: MIDI data displayed in a piano roll.

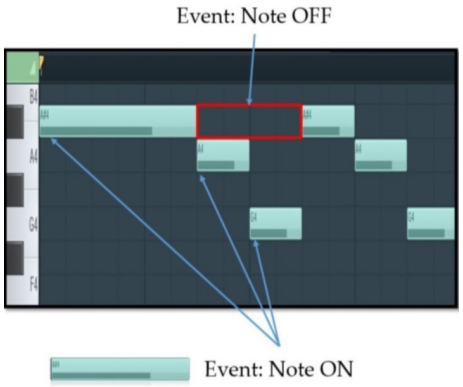
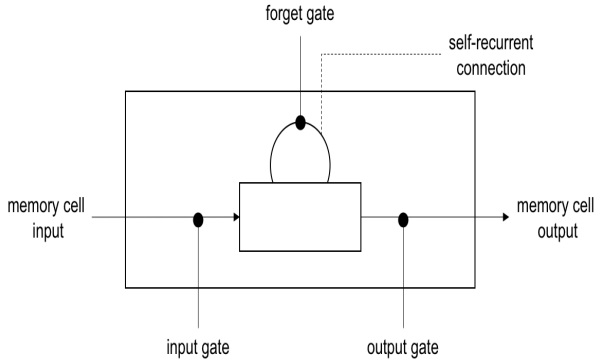


Figure 2: Simplified structure of LSTM node.



A monophonic texture consists of one layer, a melodic line with no accompaniment. In monophonic music, also known as monophony, only one note is played at a time. Monophony is not defined by the number of instruments, but by the melodic arrangement. As long as the instruments play the exact same melodic arrangement simultaneously, a piece can classify as monophonic.

A polyphonic texture consists of two or more layers that are independent of each other. Polyphonic music, also called polyphony or counterpoint, has independent melodic lines that are played simultaneously. Staggering the same melodic line also classifies a piece as polyphonic.

In terms of generating by a computer program, monophonic music is much simpler to generate. This is due to monophony only having one layer, the melodic line. Only pitch and duration are considered. Polyphonic music is much harder to generate due to there being multiple notes being played at any given time. Along with pitch and duration, polyphony also considers coincidence of notes.

2.2.3 MIDI

Musical Instrument Digital Interface, or MIDI, is a protocol that permits electronic musical instruments, computers, and various other hardware to communicate. MIDI itself does not create sounds, but instead carries a sequence of messages and instructions called events that represent note information such as

- Note On: the pitch of the pressed key represented with a value from 0 to 127
- Note Off: the time when a pressed MIDI key is released

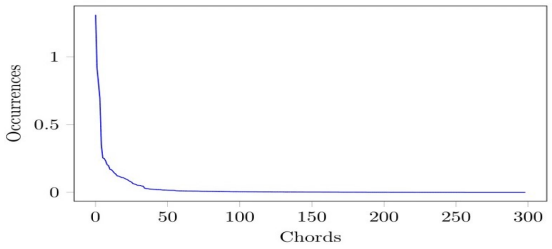
These sequences of messages are interpreted by a MIDI device, such as a MIDI keyboard, to produce sound. MIDI contains 128 different pitches, each labeled by the pitch the octave is in. *C1* denotes the pitch of C in octave 1. MIDI contains the range from A0 to G9. The entire range can be seen in what is called a *Piano Roll*. A piano roll is a way of visualizing notes. Within a piano roll, MIDI data and parameters such as the ones mentioned here can be edited. Figure 1 displays events of notes.

3. JAMBOT

It has been stated that feed-forward neural networks are not effective at modeling sequences. This is because nodes in the network only consider their current input and not input from previous iterations. Music is sequential, and to model it effectively we must be able to model patterns from sequences of input data. Recurrent neural networks can model sequences effectively but are difficult to train. The vanishing and exploding gradient problems discussed in Section 2.1.3 make learning long-term temporal dependencies difficult.

A way to avoid vanishing and exploding gradients is by using long short-term memory, or LSTM networks. LSTM networks are a variant of recurrent neural networks discussed in Section 2.1.1, that introduce memory cells with a gating architecture. The memory cells consist of an input gate, a "forget" gate, and an output gate that regulate the flow of information in and out of the cell [6]. The "forget" gate is responsible for removing information that is no longer needed from the cell state. It takes two inputs, the hidden state from the previous cell and the input at that timestep. These inputs are multiplied by their respective weight and run through a sigmoid function, which outputs a vector with values ranging from 0 to 1 that correspond to values in the cell state. If the result is a 0 for a value in the cell state, the "forget" gate removes that information. If the result is a 1, the information is kept. That vector is then multiplied to the cell state. The input gate is responsible for adding new information to the cell state. Like the "forget" gate, it first determines what information needs to be added to the cell state by taking the hidden state from the previous cell and input at that timestep and running them through through a sigmoid function. This acts as a filter for information. Those same inputs are run through a hyperbolic tangent function, a non-linear activation function with a range from -1 to 1, which then outputs a vector that contains all the possible values that can be added to the cell state. The output of the sigmoid function is multiplied by the output vector from the hyperbolic tangent function which is then added to cell state via addition. This process ensures that only important information is added to cell state. The output gate is responsible for outputting useful information from the current cell state. The current cell state is run through a hyperbolic tangent function, outputting a vector with values within the range of -1 and 1. The cell state from the previous cell and input at that timestep are run through a sigmoid function and output a vector, which is then multiplied by the vector outputted from the hyperbolic tangent. The result is sent as the output and to the hidden state of the next cell. Figure 2 shows a simplified LSTM node structure.

Figure 3: Chord occurrences in shifted dataset.



The JamBot [2] framework is composed of two LSTM networks referred to as the Chord LSTM and the Polyphonic LSTM. The Chord LSTM outputs the probabilities of chords which is used as an input for the Polyphonic LSTM to use as a guide for generating polyphony.

3.1 Dataset

The Lakh MIDI dataset is a collection of over 100,000 unique MIDI files. This dataset is used for training both LSTM networks. To train the networks efficiently we consider that there are thirty-six different scales and twelve different keys music can be in. This results in 432 unique ways music can be learned in. Instead of letting the networks attempt to learn each way, we only consider music in major and relative minor scales since there is only one key difference between the two scales. A subset is created called the Shifted dataset which contains all MIDI files from the original Lakh MIDI dataset that are in the major and relative minor scales, totaling over 86,000 MIDI files. To simplify training even more, all MIDI data in the shifted dataset is transposed to the same key, C major via piano roll. These adjustments not only simplify training but also act as precautionary steps for avoiding overfitting issues that may be caused by lack of data per key [2].

3.2 Chord LSTM

3.2.1 Data Representation

An automated process extracts chords from the dataset by computing a histogram of all twelve notes played in the time span of one measure. The time span of a measure is chosen as chords tend to change every measure. The three most occurring notes in every measure form a chord. Chords can contain more than three notes, but in this context we only consider chords to be three notes. From this process, 300 unique chords are found in the dataset. Of those, fifty chords appear frequently. A separate study done on 414,059 songs ranging from 1958 and 1991 to detect chords conducted by Burgoyne et al. [3] lists their results of most occurring chords found. In both datasets, the top ten occurring chords coincide well, validating the chord extraction process [2]. Figure 3 shows the number of chords found and the occurrences of them.

Brunner et al. [2] use a technique from natural language processing to represent the extracted chords. The fifty most occurring chords are given an integer ID and stored in a dictionary while the remaining 250 are given an ID of an unknown tag as there are not enough occurrences of those chords for the network to learn anything meaningful. The network does not know the notes that make up the chords, it only sees the IDs. In order to be able to feed these chords into the network they are encoded as vectors using *one-hot encoding*, a process that converts categorical features or variables into numerical variables represented by a vector where all the elements of the vector are 0 except one which has a value of 1. In this context, each chord/ID pair is represented with a vector with a 1 for its chord and a 0 for all other forty-nine chords. The one-hot vectors are denoted by x_{chord}.

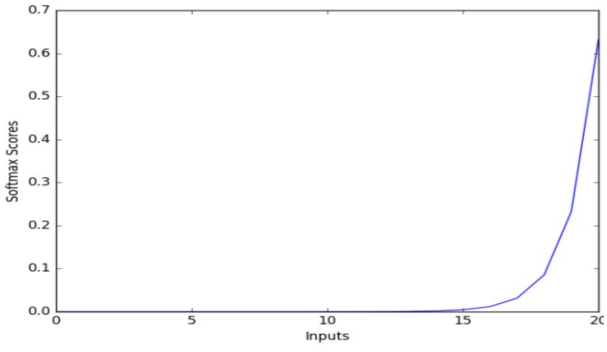
3.2.2 Structure

The first layer of the network utilizes a technique from natural language processing called *word embeddings*. Word embeddings map objects from a dictionary like the one mentioned in Section 3.2.1 to real number vectors. Consider the following sentences.

- The woman eats an apple.
- The man eats an orange.

In the vector space the words apple and orange are close together because they are similar within their usage context. Similarly, the words man and woman are close together as well, but man and woman are not close to apple nor orange because they are not similar within the usage context. These embeddings are not fixed but learned from data [2]. Mikolov et al. [4] demonstrate that embedded vector space can capture relationships without knowing any information about the content. It is applicable here, as the goal for the network is to learn a meaningful representation of the chords. An embedding matrix is denoted by W_{embed}. It produces closer results for samples that appear in the same usage context and farther results otherwise. W_{embed} consists of learnable parameters that will be adjusted during training. The W_{embed} is multiplied by one-hot vectors x_{chord} to create a 10-dimensional embedded chord vector denoted by x_{embed} [2]. This will be the input for the Chord LSTM.

Following the first layer are the hidden layers. Brunner et al. [2] do not specify the number of hidden layers. We only know that there are 256 hidden cells. This could mean there is one hidden layer which is iterated over multiple times. Following the hidden layer is the output layer. Softmax, a commonly used activation function in the output layer is used as the output activation function. The softmax function compresses the outputs into a range of 0 and 1 and divides each output such that the sum of the outputs equal 1, effectively becoming a categorical probability distribution that describes the probability of any case to be true. The output of the Chord LSTM is a vector that contains the probabilities for all chords [2].



3.2.3 Training

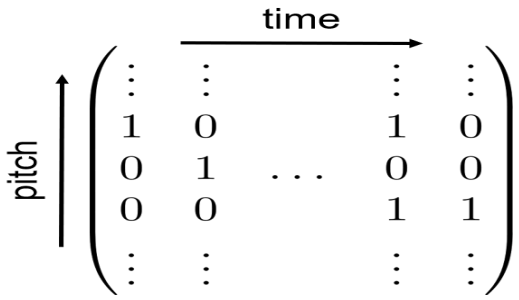
The network is trained with the extracted chords from the shifted dataset for a total of four epochs. An epoch is a unit of measurement representing one full training cycle. A training cycle is considered full once every sample from the training set is run though the network.

To generate a chord progression, a seed is fed into the network. The first chord is generated by sampling the output vector with *temperature*, a hyper-parameter with a range of 0 to 1 used to control randomness of predictions. Temperature scales the probabilities which are chord probabilities in this context, and then applies a softmax function to normalize them. Sampling with a high temperature such as 1 causes all samples to have nearly the same probability, while sampling with low temperature such as 0 keeps the probabilities the same. In this context, temperature controls the randomness of chords to be generated. A temperature value of 0 results in the same chord always being generated. A temperature value of 1 results in a very different variety of chords generated. The generated chord is then fed back into the network and the following chord is generated by repeating the same procedure. The result of this is a chord progression.

3.3 Polyphonic LSTM

3.3.1 Data Representation

All MIDI data is represented via piano roll. Brunner et. al. [2] begin by dividing every measure into eight timesteps. A vector represents notes played at each timestep. An entry of 1 means the note was played and an entry of 0 means the note was not played. The vector length is the total number of notes.



3.3.2 Structure

Polyphonic LSTM uses the Chord LSTM chord probabilities output as input along with other features. The input vector for the Polyphonic LSTM consists of the probabilities of chords to be generated, the probabilities of the following chords to be generated, the piano roll vector, and a counter. The probabilities of chords and following chords come from the Chord LSTM. They are included to give the generated polyphony structural guidance. The first chord is used as a basis for the structure of the polyphony. In Section 3.2.1 we mention that the network does not know the notes that form the chords. Brunner et. al. [2] hope that the network learned meaningful embeddings so it can identify patterns between notes and chords.

A way to give the first generated notes a direction, so to speak, is by including the probabilities of the following chord to be generated. Pleasant music tends to have moments of tension and resolution. Some tension or resolution can be present when chords change. In polyphonic music, the melody, the generated polyphony in this context, leads the composition to those moments. By giving the generated polyphony a sense of direction for notes to generate, the network can model those moments thus generating more pleasing polyphonic music. To model it effectively the network needs to know when the chord change is coming. A binary counter is used to keep track of timesteps. In this context, a chord lasts eight timesteps. When the counter reaches timestep seven, the network knows that a change is to come next.

These input vectors are fed into the Polyphonic LSTM. Brunner et al. [2] do not mention the number of hidden layers. We only know there are 512 hidden nodes. Following the hidden layers is the output layer. The Polyphonic LSTM outputs a vector containing the probabilities of notes to be generated at the next timestep given the notes that were generated previously.

3.3.3 Training

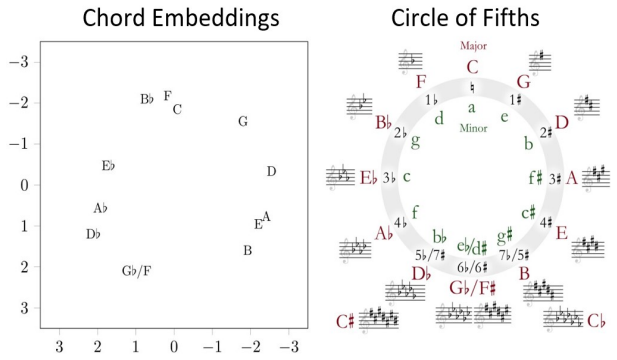
The network is trained with 10,000 MIDI files from the shifted dataset for four epochs. To generate polyphony, a seed is fed into the network. The seed consists of the piano roll vector and chord vector. Notes to be generated are sampled with hyper-parameter temperature, which we discuss in Section 3.2.3 from the Polyphonic LSTM output vector. In Section 2.2.2 we explain that polyphonic music can have multiple notes playing a time. To achieve this, Brunner et. al. [2] sample each note independently. Sampling notes independently prevents the probabilities of all other notes to be generated from changing. That way if two or more notes can be played at the same time, they will. To avoid large number of notes playing at the same time a limit is implemented.

4. RESULTS

Principal component analysis or PCA is a technique that makes data easy to visualize. Brunner et. al. [2] use PCA to visualize chord embeddings trained on the shifted dataset in two dimensions. Instead of plotting IDs of chords, the notes that make up the chords are plotted for the fifteen most occurring chords. From the visualization, Brunner et. al. [2] note that chords containing common notes are close together in the vector space. When applying the same technique to the original dataset and plotting chords instead of individual notes, the result resembles the circle of fifths, a guide used for understanding relationship between keys and chords. Figure 4 shows the chord embeddings in two dimensional space and the circle of fifths. From the diagrams we can see the resemblance in chord placement. Thus the network is able to understand principles of music theory without actual rules being implemented.

After listening to music generated by JamBot [2], we can hear clear long term structure. The transitions within the songs are smooth with no sudden changes. Music style is consistent throughout the songs as well. At times there are dissonant notes present. This might be caused by sampling with a high temperature. Even if the probabilities of those dissonant notes are small, sampling with a temperature value of 1 increases their probabilities of being picked. Music generated by JamBot [2] is available for listening on YouTube. A link is provided in the reference section [1].

Figure 4: Comparison between chord embeddings and the circle of fifths



5. CONCLUSION

JamBot [2] is a framework capable of generating pleasing polyphonic music. It is able to extract the circle of fifths, a guide used by many musicians for understanding relationships between keys and chords. It provides a relatively simple way of generating music. It is possible to generate music of a specific genre by training the networks with genre specific songs. Songs generated by JamBot [2] are not necessarily the final product. Because JamBot [2] generates what is essentially a MIDI file, generated songs can be edited within a piano roll to produce more complex songs or used as inspiration for creating a much more unique piece.

6. ACKNOWLEDGMENTS

I would like to thank Elena Machkasova, my professor and adviser for senior seminar, and my alumnus reviewer, Dan Stelljes, for all your guidance and feedback.

7. REFERENCES

[1] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger. Jambot generated music. `https://www.youtube.com/channel/UCQbE9vfbYycK4DZpHoZKcSw/videos?shelf_id=0&sort=dd&view=0`, 2017. [Online; accessed 12-November-2018].

[2] 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)*, pages 519–526, 2017.

[3] J. A. Burgoyne, J. Wild, and I. Fujinaga. An expert ground truth set for audio chord recognition and music analysis. *2011 The 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 633–638, 2011.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 2013.

[5] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–1318, 2013.

[6] Wikipedia. Long short-term memory, 2018. [Online; accessed 09-October-2018].

[7] Wikipedia. Musikalisches Würfelspiel, 2018. [Online; accessed 12-November-2018].