JACOBS
UNIVERSITY

Echo State Networks: Music Accompaniment by Prediction

by

Tomas Pllaha

Bachelor Thesis in Computer Science

Prof.Dr. Herbert Jaeger
Name and title of the supervisor

Date of Submission: May 11, 2014

Jacobs University Bremen — School of Engineering and Science

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature   Place, Date

Abstract

This thesis describes and outlines the results of an experiment that focused on exploring the use of Echo State Networks (ESNs) for a Music Accompaniment task. The system was trained to “listen” to a leading (human-composed) melody and play along by generating an accompanying track in real time. The leading melodies that were selected for training and testing were composed for either guitar or piano, and the system was trained to accompany these melodies on the bass. An attempt was made to give the network as much freedom as possible, to imitate the real life conditions of a human musician during an improvisation session. We analyzed the extent of the network’s “creativity” as well as several other aspects of its performance.

Recurrent Neural Networks (RNNs) and other Machine Learning techniques have been previously used for music generation tasks. The novelty of the approach presented in this thesis thus lies in using ESNs, which are generally much easier to train and faster than other RNNs. They have been found to be very suitable for prediction of time series (data points - measured typically in equally spaced time intervals), and that is exactly what music is - a sequence of tones measured at different points in time.

While several improvements can be made to the system’s current state, our results suggest that it is possible for Echo State Networks to imitate creative behavior in music. This type of Recurrent Neural Network remains not fully exploited for the music generation task. However we observed an ability to generate music that is both original and correct. The use of such a system would be in helping composers create music by generating improvised tracks. Moreover, tackling music generation tasks can be useful in furthering research about creative AI.

Contents

1 Introduction

Several methods have been explored to tackle music generation tasks. Some of first methods used used to produce software able to generate music relied on algorithmic composition [1]. While these methods sometimes produced satisfactory results, their output suffered from a lack of originality. Markov Chains have also been exploited for composition of music [2]. While they produce melodies that are more random/original than those achieved by algorithmic composition, they also suffer from drawbacks, the largest being that the structure of music violates the Markov hypothesis: that the output at each step only depends on the output of the previous step. An unconventional computing approach to tackle the music generation problem made use of Memristors [3], motivated by their time dependecy property and similarities to neural networks. Memristors, however, are very difficult to train and one of their greatest disadvantages is that it is not known whether they are deterministic or non-deterministic in nature. RNNs have also been used several times in the past. CONCERT [4] is an example of a network that achieved satisfying results in 1994. This particular RNN had a complex structure, which was based on human perception.

Echo State Networks represent a type of Recurrent Neural Network. As shown in [5], they are significantly easier to train and converge faster. They consist of a set of input neurons, a reservoir of randomly generated, recurrent neurons, and a set output neurons. The recurrent reservoir enables ESNs to learn to imitate arbitrary dynamical systems. While in classical RNNs all weights need to be updated, the novelty of ESNs lies in the fact that the output weights are the only weights that need to be learned (See Figure 1). The reservoir weights are random and fixed.

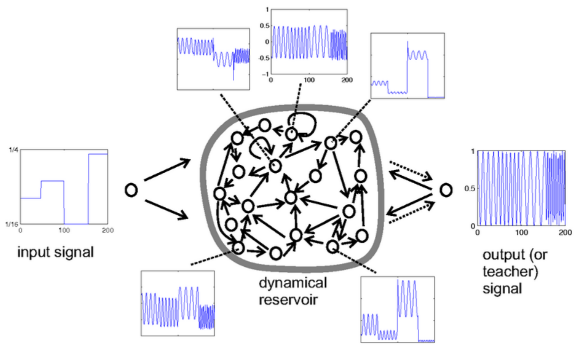


Figure 1: The basic schema of an ESN, illustrated with a tuneable frequency generator task. Solid arrows indicate fixed, random connections; dotted arrows trainable connections source: http://www.scholarpedia.org/article/Echo_state_network

Since the output of an ESN is just a linear combination of the reservoir units, the learning task is a simple linear regression. However, ridge regression is the most commonly used method of training ESNs, as it tends to yield more stable solutions [6]. The resevoir random weights can be easily scaled to ensure the convergence of the system [7]. Because the learning task is performed with increased efficiency by ESNs, they are much faster than other RNNs. The trade-off for this gain in speed is the increased size of the network (a very large reservoir is usually required).

Therefore, the advantages that ESNs offer over other methods for tackling the music generation task can be identified as:

1. They are easy to use and fast to train.

2. Their Echo State Property [8] makes them very suitable for this particular task. In ESNs, the output at a given step depends on the recent past. Events in the more recent past affect the output at the present step more than those that occurred earlier. Intuitively, this is very similar to the structure of music.

3. The structure of the network is simple and pre-determined, reducing the need for construction of a complex network architecture.

Previous results have shown that while software is able to generate correct music, it is difficult to build a system, which produces melodies that are both original and musically pleasing. Nonetheless, satisfying results have been achieved in accompaniment systems (e.g.: MySong [9], the Continuator [10]) using Hidden Markov Models (HMMs). Intuition suggests that this could be a result of the human creativity factor put in the system through the leading track. Throughout this experiment, we investigated the behavior of an Echo State Network designed to mimic human improvisation as closely as possible. Section 2 contains detailed information about the research problem and motivation of research. Section 3 describes the experiments that were conducted and classifies them. Section 4 outlines the results of these experiments. Finally, section 5 concludes with a summary of the main findings of this thesis, and discusses potential focus areas of future research on this topic.

2 Statement and Motivation of Research

Recurrent Neural Networks are a biologically inspired model. Each unit is analogous to neurons and the connections between units are analogous to axons and synapses, which conduct impulses among neurons. As such, it seems intuitively sensible to utilize them for tasks that require simulated creativity, such as music generation. However, classical RNNs require significant computational power and an advanced level of task-specific expertise is required to build robust RNN architectures. These requirements were reduced dramatically by the introduction of Echo State Networks and Liquid State Machines [11], which partially explains the attention they have received in the field of Machine Learning. This section starts with a formal description of Echo State Networks. Subsection 2.2 gives more detailed information about the research, and subsection 2.3 concludes this section with a list of the research questions that were addressed throughout this investigation.

2.1 Echo State Networks

As mentioned in the introduction, ESNs consist of an input vector $\mathbf{u}$ of size K, a reservoir (random vector) $\mathbf{x}$ of size N, and an output vector $\mathbf{y}$ of size L. Let the input to reservoir connection weights be stored a matrix W^{in} of size $N \times K$. Let W be the $N \times N$ matrix of the reservoir connection weights and W^{out} the $L \times K + N$ matrix of output connection weights. Furthermore let W^{fb} be the $N \times L$ output feedback matrix and f a sigmoid function (typically the logistic function or $tanh$). Then the network is governed by the following state update equation [5]

$$\mathbf{x}(n+1) = f(W\mathbf{x}(n) + W^{in}\mathbf{u}(n+1) + W^{fb}\mathbf{y}(n)) \tag{1}$$

In our music accompaniment task, no feedback is required, so W^{fb} can be nulled. The output is obtained by the following equation:

$$\mathbf{y}(n) = g(W^{out}[\mathbf{x}(n); \mathbf{u}(n)]) \tag{2}$$

In the above equation g is the output activation function (typically identity or a sigmoid function), and $[\mathbf{x}(n); \mathbf{u}(n)]$ is the concatenation of $\mathbf{x}(n)$ and $\mathbf{u}(n)$.

The output matrix W^{out} is learned by performing regression on the extended reservoir states (see below), after the network is run with the training data. The desired outputs for all timesteps are stored in a matrix D of size $n_{max} \times L$, where n_{max} is the length of the training sequence. The extended reservoir states $[\mathbf{x}; \mathbf{u}]$ are stored in a matrix S of size $n_{max} \times N + K$. One could use the Wiener-Hopf solution: $W^{out} = R^{-1}P$, or Tikhonov Regularization: $W^{out} = (R + \alpha^2 I)^{-1}P$, where $R = S'S$ is the correlation matrix of the extended reservoir states (the prime denotes the transpose), $P = S'D$ is the cross correlation matrix of the extended network states and the desired outputs, α^2 is a non-negative number (the larger, the stronger the smoothing effect), and I is the identity matrix.[12] Tikhonov Regularization is the recommended standard, because it yields stable solutions.[6]

2.2 Statement of Research

In this subsection we will explain the purpose of this guided research in further detail and argue the properties of Echo State Networks that make them suitable for this particular task. Furthermore we will try to justify the research questions, which follow in section 2.3.

The current state-of-the-art systems that tackle the same task use either HMMs or classic RNNs. As stated in the previous sections, ESNs are a biologically inspired model and converge faster than classic RNNs. They have also been proven to perform particularly well on time series prediction tasks. These reasons led us to formulate the idea of utilizing them for the music accompaniment task. This research focuses mainly on testing the extent of "creativity" ESNs can display when generating accompanying music. [1] The long-term goal is to build systems that can generate melodies that are satisfying for the human listener, while at the same time being original and consistent with the musical context established by the leading track. For this reason, the imposed restrictions were very limited.

One of the defining properties of ESNs is the so called Echo State Property, which is easily enforced by scaling the reservoir weight matrix to obtain a spectral radius smaller

[1] Due to its subjective nature, the notion of creativity is used loosely in this context. An in-depth discussion of this issue would stray too far from the subject matter.

than 1. [8] Simply put, the property states that the effect of the initial state of the reservoir on the output vanishes out after sufficiently long runs. In other words, after sufficiently long runs, the output is a function of the input vectors only. This also means that recent input affects the output more than earlier input. This property seems very suitable for music generation tasks, because in general, songs can change a lot as they progress, but sequences that are close to each other are generally consistent with the same musical context. Moreover, this property is especially suitable for the instrumental accompaniment task, because such system could also be used for improvisations. If the style of the leading track changes, the network should be able to respond and eventually forget the previous style (The network is state forgetting [8]). For this reason, the response of the network to changes in the musical context was also investigated.

This Short Term Memory [7] of ESNs is of great use for the music generation task. The system cannot have any knowledge of the future, so it has to rely on its knowledge of the recent past. During training, at every time step, the next note of the accompanying track is forced in the output. In other words, the network is trained to always predict the next note on the accompanying track by taking as input the current note on the leading track. It currently supports only one line of melody per track and therefore cannot play chords.

In the following subsection, we summarize this statement of research by listing all the questions that were addressed throughout the investigation.

2.3 Research Questions

Throughout this guided research the following research questions were addressed:

1. How can ESNs be used as an instrumental accompaniment system?
2. What are the network parameters that yield the best results?
3. What way of representing input yields the best performance?
4. How does the performance of the system depend on the input?

3 Conducted Experiments

This section describes the experiments that were conducted to address the research questions. We used two different sets of midi files to train the network. The tools csvmidi and midicsv [13] were used to convert between csv and midi formats. One of the training sets consisted of 16 rock songs (approximately 20000 time steps) and the other one consisted of 70 Beatles' songs (approximately 90000 time steps). Approximately three quarters of the datasets were used for training and a quarter was used for testing. A full list of the songs used, as well as a link to the source code and some sample files can be found in the appendix.

In the first subsection, we start with an explanation of the mathematical representation that was chosen to represent musical notes. In subsection 3.2, we describe the architecture of the network and the training and testing procedures that were implemented. We continue with a brief description of the experiments that were run to determine the most suitable network parameters in subsection 3.3. In the 4th and last subsection we give a classification and description of the different types of experiments that were conducted in order to answer the research questions.

3.1 Representation of Musical Notes

The way we choose to represent the musical notes for this task plays a significant role in the performance of the network. Although the input is transformed in a non-linear fashion inside the reservoir and Echo State Networks converge fast, the more knowledge we put in the system, the easier the prediction task will become. Obviously, the representation of input is a way adding knowledge to the system; this knowledge is crucial, because of the complexity of the task we are dealing with. We extract information about notes from midi files and we represent notes with two main features: pitch and duration.

3.1.1 Representation of Pitch

The songs used for training were originally in different keys. With the intention of avoiding the added complexity of supporting multiple keys, all major songs were transposed to C major, and all minor ones to A minor. Moreover, to allow for a succesful representation of all notes in the training set, representation of five octaves is supported. (C1,C1#, D1, .. C6, where Xn stands for the note X on the nth octave). As it has already been mentioned, the network only supports one voice per track and therefore chords are not supported. Whenever chords appeared on the training set, only the highest note was used.

One of the ways of representing pitch would be through a vector of size $61 = 5\times(7+5)+1$, where all elements are set to 0, except for the one that corresponds to the note we wish to represent, which is set to 1. For example C1 would be [1 0 0 ...0], C1# would be [0 1 0 0...0], D1 would be [0 0 1 0 ...0] and so on. The problem with this approach is that the differences between all notes appear to be equal, so no knowledge about the value of each pitch is entered in the network. (One could measure the difference between two pitches represented this way either by Euclidean distance or by the angle between the vectors - both of these measures are equal for any two different pitches in this representational space).

Another approach would be to represent pitch by one single value, an integer from 0 to 61, where 0 stands for rest, 1 stands for C1, 2 for C1# and so on. Every number greater than 1 represents the positive distance of that note from C1 in semi-tones. This representation differentiates between notes that are close to each other and those that are not (notes that are close to each other have a smaller mathematical difference). However, it does not otherwise represent similarity among different notes. Ideally, the same notes played on different octaves should have relatively small distances from each other, although their absolute pitches are not neccessarily close. Therefore the representation used by CONCERT [4] was implemented.

The pitch of each note was encoded as a 5-dimensional vector. In this vector, the first unit has a value proportional to the logarithm of the absolute pitch of the note. The second and third units are (x, y) coordinates of the position of the note in the chroma circle. Similarly, the fourth and fifth units code the (x, y) coordinates of the note in the circle of fifths. This representation is based on Shepard's theory of generalization [14], according to which the perceived similarity of two items decreases exponentially with the distance between them in an internal or 'psychological' representational space [4].

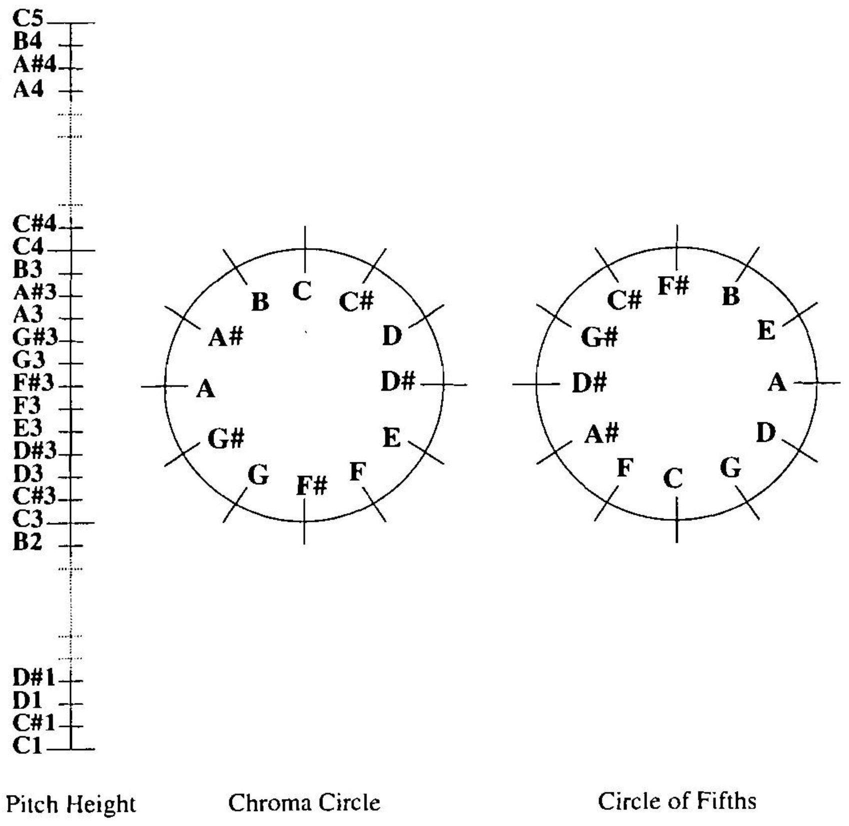


Figure 2: Shepard's (1982) pitch representation [4]

The logarithmic transform of absolute pitch places tonal half-steps at equal distances from one another along the pitch axis. In the chroma circle, neighboring pitches are a tonal half-step apart. In the circle of fifths, the perfect fifth of a pitch is the pitch that follows immediately counterclockwise [4].

The relative importance of each of these three components in determining the similarity of two different notes can be adjusted by changing the diameters of the chroma circle and the circle of fifths. In our experiments, we followed Shepard's [15] recommendation by giving the same diameter to both circles and making it equal to the length of one octave on the pitch axis.

In the following table we list the Euclidean Norms of the differences between C1 and each note between C1# and C3 in this five-dimensional space.

Note	Distance from C1
C1#	2.0069
D1	1.4530
D1#	2.0616
E1	2.5386
F1	2.1667
F1#	3.0000
G1	2.3154
G1#	2.7889
A1	2.5000
A1#	2.1858
B1	2.7131
C2	2.0000
C2#	2.9486
D2	2.7285
D2#	3.2016
E2	3.6209
F2	3.4681
F2#	4.1231
G2	3.7454
G2#	4.1366
A2	4.0311
A2#	3.9299
B2	4.3237
C3	4.0000

Table 1 : Distance between C1 and each note in [C1# : C3]

3.1.2 Representation of Duration

To allow for a correct representation of all songs in the training set, 16 note durations were supported (all durations between $\frac{1}{16}$ and $\frac{16}{16}$ inclusive). The duration of each note was encoded with only one duration height dimension. The duration height is proportional to the logarithm of the duration. This logarithmic transformation follows the general psychophysical law (Fechner's law) relating stimulus intensity to perceived sensation[4]. It was not necessary to map durations to a higher dimensional space, because the durations produced by the network using this representation were already precise. The average distance between the durations of the network output and those of the testing accompanying track at the respective time steps was approximately 1 in nearly all runs.

3.1.3 Representation of Beat

In most experiments the network was also fed a beat. The beat was taken from the percussion instruments of the training and testing set and was encoded with a single unit. This unit had a value of 1 whenever a percussion instrument was hit, and 0 at any other time.

3.2 Network Design

In this subsection we give a description of the network architecture used for this study, leading up to the next subsection, which lists the network parameters. The network was designed to take the input notes in the representations described in the previous subsections. The output at each step during testing, is a vector of probabilities of selecting each note (and duration) as the next one on the accompanying track. This allowed for testing the network with deterministic as well as non-deterministic runs. This procedure described in further detail below.

The input of the network used for this study consists of 14 units, whereby one unit is used as a bias input, another is used to represent the beat, and six units are used to represent each of the notes on the two tracks. 5 out of these 6 units encoded the pitch in the five-dimensional space described earlier in this section and the other was used to represent the duration of the note.

The output consisted of $61 + 16 = 77$ units. The first 61 units were used to represent the next pitch and the other 16 units were used to represent the next duration. Below we give a description of the training and testing procedures, which will also clarify why the notes were not encoded in the output neurons in the same way as they were in the input neurons.

During training, at every step n, as input we feed the n^{th} note on the accompanying track, the n^{th} note on the leading track, and the n^{th} "beat value" (1 if a percussion instrument is hit at time n, 0 otherwise). In the input vector, we represent notes in the 6-dimensional space described above (5 dimensions for pitch, 1 for duration). In the output we force the $n{+}1^{th}$ note of the accompanying track at every time step n. By doing so, we teach the network to predict the next note on the accompanying track. In the output, however, we use the most simple representation of notes, whereby pitch is encoded in 61 neurons, each of which corresponds to a musical note (this form of representing notes was described in further detail in section 3.1.1). In the same manner, we code the output duration with 16 neurons. We chose this form of represetation of notes in the output neurons because we want to be able to interpret the network output as a probability vector. Therefore, at every time step during testing, each neuron should have a value proportional to the probability of selecting the respective note at that time.

During testing, at every step n, as input we feed the n^{th} note of the leading track and the n^{th} beat value. The network produces an output, which we use to select the next note on the accompanying track. The selected note is fed back to the network as the accompanying track input at step $n+1$, together with the $n+1^{th}$ note on the leading track and the $n+1^{th}$ beat value. The process is repeated until the end of the testing sequence.

A careful reader may notice that using this procedure, we immediately encounter a problem. Let us assume that at time n the note on the leading track has a duration of 2 and the note on the accompanying track has a duration of 4. The network produces the next note of the accompanying track (i.e.: the note at time step $n + 1$). However, this note should not be fed back to the system at time $n + 1$, because at that time the previous accompanying note (the one with duration 4) would still be playing, which would cause a conflict. In order to solve this problem, every input of duration d is fed to the network for d consecutive time steps. One could say that this eliminates the need for coding duration in the first place, because the duration is implicitly given by the number of consecutive time steps in which the note is entered in the network. However, by adding an extra neuron to code duration, we make this implicit information explictit, thus reducing the complexity of the learning task. Furthermore, by coding duration with an extra input neuron, we can distinguish between two consecutive $C1$ notes of duration 2 and one $C1$ note of duration 4.

In the following subsection, we describe the network parameters and the evaluation methods used to determine them.

3.3 Network Parameters

In order to determine the best network parameters we considered the following measures:

1. The mean and standard deviation of the network outputs versus the mean and standard deviation of the accompanying sequence used for testing,
2. The mean distance between the network outputs and the notes on the accompanying track (using the six-dimensional representation of notes)
3. The logarithm of the product of probabilities of selecting each note of the testing accompanying track at the right time step.

The experiment started with a reservoir of 400 neurons for the small training set. This reservoir size produced melodies that were too monotone and contained long sequences of the same note. The standard deviation of the network outputs was between 2 and 3, while the standard deviation of the notes on the accompanying track of the testing set was approximately 15. We continued to increase the network size, finally reaching a reservoir size of 1000 for the small training set and a reservoir size of 2000 for the large training set.

Henceforth, whenever the training set used is not specified, we are referring to the large training set (containing Beatles' songs, as this is the training set most experiments were performed on.

After manually experimenting with several sets of parameters, we decided on a reservoir with 2000 neurons, an internal reservoir weight matrix with a spectral radius equal to 0.5, input weights ranging between -0.7 and 0.7 and no output connections. The output connections are not needed for this prediction task, because the output at step n is fed back to the network as input at step $n + 1$, and therefore the information contained in the output does not escape the reservoir.

3.4 Types of Experiments

To conclude this section, we list the types of experiments that were conducted, leading up to the following section, which describes the experimental results.

We classify our experiments based on their purpose, i.e: based on what they were intended to determine.

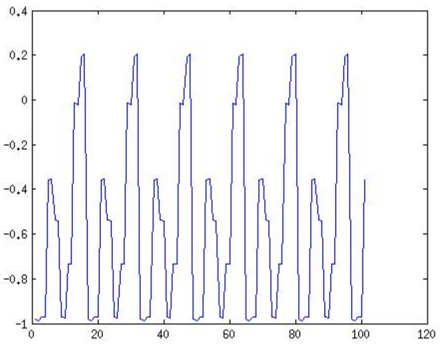
1. *Deterministic vs. Non-Deterministic choice*. We can select the next accompanying note deterministically, by selecting the one with the highest value on the corresponding output neuron. Another option is to transform the output into a probability vector, by setting all the negative entries to 0, raising all the remaining entries to the power of F and normalizing them so that they sum up to 1. (here F is a positive "favor factor", used to favor higher output values over proportionality [7]. If F is too low, the output will be too random. If F is too high, the output will consist only of sequences that are already in the training data and the variability will be too low.) Then we can select the next note by a weighted random draw where the weights are given by the updated output vector. We observed that whenever the output was selected deterministically, noise needed to be added to the reservoir during training, to increase the standard deviation of the outputed notes. This was not neccessary when selecting the output non-deterministically.

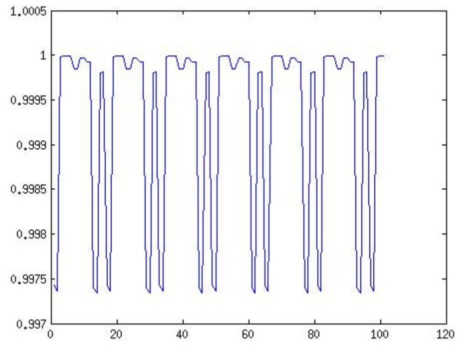
2. *The Impact of Beat*. We experimented with the input connection weights relating the "beat" neuron to the reservoir units. This highly affected the precision of output durations.

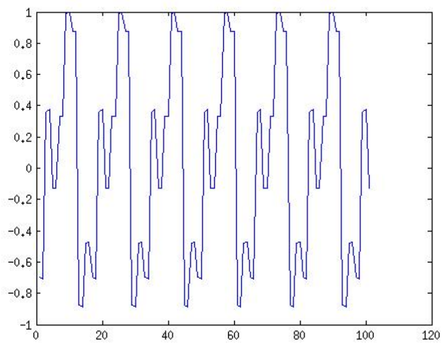
3. *The Effect of Periodic Input*. We observed that the network produced much better results when the input melody was periodic (especially when the period was not too long). More detail about this effect is given in the following section.

4 Experimental Results

This section outlines the results of this guided research. The results are grouped in the same categories as the experiments listed in the previous section. We start by giving plots of 3 randomly selected reservoir units over a 100 step sequence.







4.1 Deterministic vs. Non-Deterministic choice

The network was tested in deterministic as well as non-deterministic runs. The observed behavior was that during deterministic runs, the network would "get stuck" in long sequences of the same note, if noise was not present. Noise played a significant role in the network's behavior. Too little noise ($[-10^{-5} : 10^{-5}]$) had almost no effect, whereas too much noise $[-10^{-2} : 10^{-2}]$ produced results that seemed too random. While the introduction of appropriate noise ($[-10^{3} : 10^{-3}]$) appeared to result in more reasonable variability, this variability appeared to be "fake", in the sense that subsequent notes were not in harmony with one another.

Therefore non-deterministic selection of notes was used to increase the variability of notes. In this case, the favor factor F played a significant role in the network's performance. $F = 1$ generated almost random output, while $F = 4$ resulted in a behavior that was very similar to the one observed when selecting notes deterministically. $F = 2$ and $F = 3$ produced similarly reasonable results. The main difference between choosing $F = 2$ and choosing $F = 3$ was that the former produced more diverse output, at the cost of increased inconsistency.

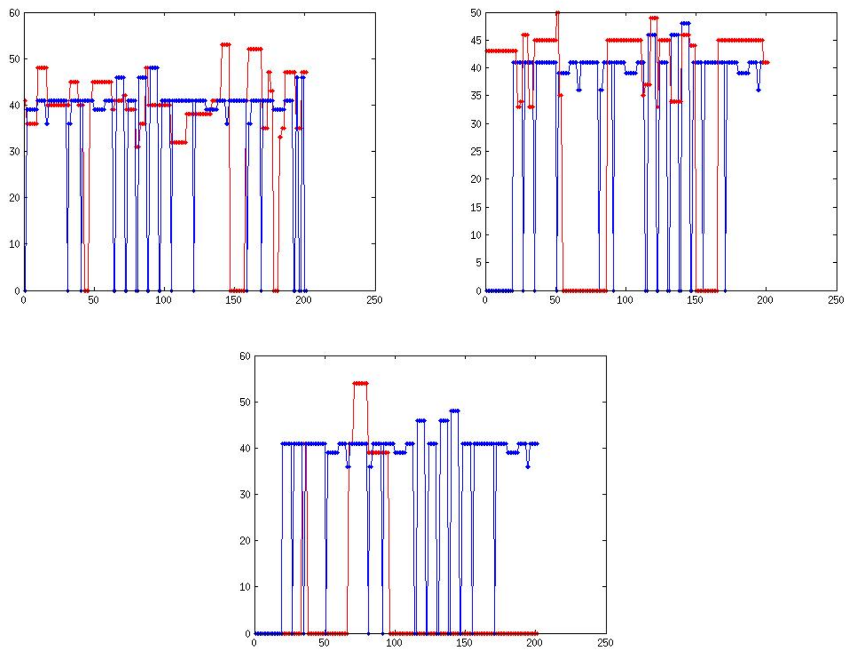


Figure 3: 200 step runs ploted agains the testing data. (Red: network output. Blue: real accompanying track. top-left: $F = 2$, top-right: $F = 3$, bottom:$F = 4$)

In the optimal case, the mean of the network-produced output was $m = 0.84M$ where M is the mean of the notes in the testing accompanying track. The standard deviation of the network-produced output was $s = 1.43S$ where S is the standard deviation of the accompanying track. The mean distance between the five-dimensional representation of network produced pitches and pitches in the testing accompanying track was reduced to to 2.13. The mean distance between the log network-produced durations and those in the testing set was reduced to 0.81

4.2 The Impact of Beat

The introduction of the “beat neuron” played a significant role in the precision of the network-produced durations. Before we used the percussion instruments to encode information about beat, the mean duration error was 1.6, which is very large, considering that this is the error after the logarithmic transformation of the durations. The connection weights of the beat neuron to the reservoir were increased in small increments from 0 to $[-1 : 1]$, which showed that the best range for the beat to reservoir connection weights is $[-0.6 : 0.6]$.

4.3 The Effect of Periodic Input

Although the network generally produced very monotone melodies when run deterministically, a particularly good behavior was observed when the input leading melody had repetitive sequences with relatively short periods. It takes the network only about 2 period-lengths to adjust to the period of the leading melody, and once it “catches the rhythm”, it plays along with a very consistent, original and satisfying periodic melody. This suggests the need for further research on training the network with larger datasets that contain only periodic melodies. This can be justified by noting that in most improvisation sessions, periodic melodies will be played. Moreover, it should be noted that in this experiment we have trained the network to improvise an accompanying track for melodies that were composed (i.e.: not improvised), which highly increases the complexity of the learning task. Below we give a visualization of deterministic runs (with no added noise) over 2 identical periodic sequences that occurred at different times within the same song.

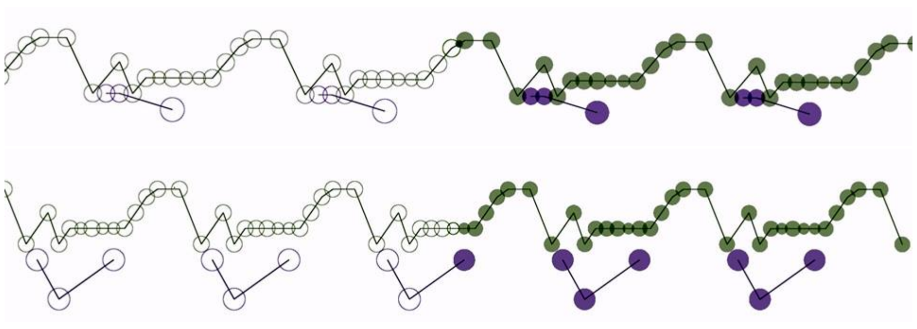


Figure 4: Two deterministic runs over identical periodic sequences occurring at different times within the same song. The purple circles (bottom) represent the bass melody produced by the network, while the green circles (top) represent the leading track (piano).

Note that the network produced different accompanying tracks for the same leading melody during deterministic runs with no added noise. Further note that both of these melodies were consistent with the leading melody. This suggests an ability of the network to produce somewhat “creative” melodies, provided that the leading melodies are simple enough.

In contrast with this behavior, we also show a run of the same network (with the same parameters) over a much more complex leading melody.

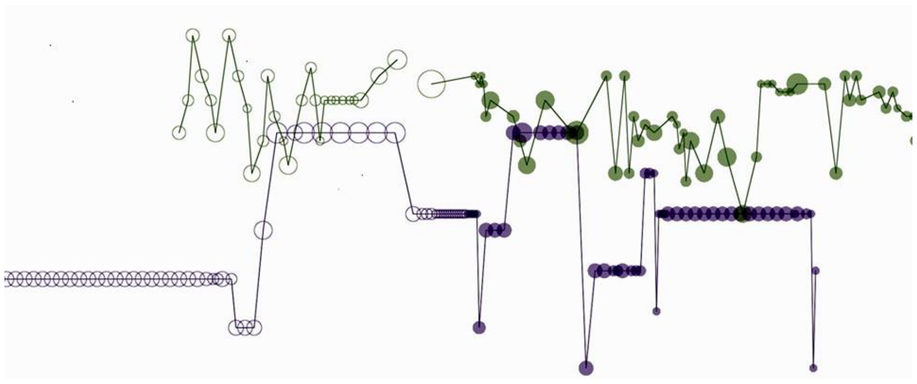


Figure 5: Deterministic run over a more complex leading melody. The purple circles (bottom) represent the bass melody produced by the network, while the green circles (top) represent the leading track (piano).

5 Conclusion and Discussion

The process for the generation of accompanying music, used in this study, was similar to real life music improvisation in three crucial ways:

1. Echo State Networks are biologically inspired.

2. The Echo State Property seems analogous to the behavior of a human musician during an improvisation session. Generally speaking, when improvising, human musicians display a tendency to eventually forget sequences of notes they produced earlier and focus only on recently played music.

3. From a human musician's perspective, improvising requires a prediction of the future.

Echo State Networks appear to be a very suitable tool for generating music by prediction. This study was able to demonstrate that under specific conditions, we can use ESNs to generate music that is "correct" and original, by providing them with as little as 70 songs for training.

It is worth noting that music generation by prediction should be considered improvisation, rather than composition. Learning to improvise pleasing music is a difficult task that takes a lot of experience, even for human musicians. Therefore in future research on this topic, a significantly larger amount of training data should be used.

It is the author's opinion that we have only scratched the surface of the "musical improvisation capabilities" of Echo State Networks and that this area offers many interesting opportunities for future research. Below, we suggest some potential improvements that could be made to the outcome of this guided research, as well as potential ways of expanding upon it.

5.1 Potential Improvements

As mentioned above, music generation by prediction should be considered analogous to music improvisation rather than music composition. Therefore, considering the complexity of training and testing data is of high importance. Further research in this area should begin by improving the classification of the training data. Starting by only using periodic melodies in larger quantities, it seems almost certain that a network producing consistently good results can be built. Afterwards, the complexity of the training data should be increased in small increments, in order to determine the extent to which Echo State Networks can be exploited for this task.

Another suggested improvement that could be made to the system built during this guided research project is to implement a better representation of beat. Often times, songs have at least two overlapping rhythms. This is achieved through different percussion instruments or different drums (e.g.: hi-hat, snare etc.). By representing the beat with only one binary unit, this information is completely lost. One possibility would be to code percussion with more dimensions, each of which would correspond to one particular instrument in the drum set.

Additionally, in order to further approximate a simulation of accompaniment by a human musician, one could consider teaching the network to also predict the next note on the leading track. This prediction could be fed back to the network through output feedback connections. This suggestion is motivated by the author's impression that when humans accompany a leading track without having any knowledge of the future, they attempt to predict subsequent notes on the leading track in order to play a suitable note in return.

Finally, if we want to build a system that consistently generates satisfying melodies, a significantly larger amount of training data should be used. An appropriate starting point could be to use c.a. 500 songs of a similar nature (e.g.: of the same genre). The song selection should also be conducted carefully, ideally by people with significant musical experience and training, to ensure that similar musical contexts are established. Due to time constraints, it was not entirely possible to implement this during the course of this guided research.

It is the author's belief, however, that during this guided research project a first step towards building software that is able to generate music that is both original and of high quality has been taken and that the feasability of using ESNs to this end has been established.

5.2 Potential Expansions

Further expansions to this system may include:

1. Adding support for more lines of melody per track to enable chord representation.

2. Adding support for more instruments (more accompanying tracks), eventually building an "orchestra" of music generating software.

6 Acknowledgements

I would like to express my most sincere gratitude to the supervisor of this guided research, Prof. Dr. Herbert Jaeger, without whose continuous support and guidance the realization of this project would not have been possible.

7 Appendix

7.1 List of Songs Used for Training and Testing

Small set
Eric Clapton - Cocaine
The Doors - Riders on The Storm
The Doors - Light My Fire
The Doors - People Are Strange
The Doors - Break On Through
The Doors - Hello I Love You
Dire Straits - Walk of Life
Dire Straits - Money For Nothing
Bon Jovi - Living on a Prayer
Deep Purple - Smoke on The Water
Nirvana - Come as You Are
Eric Clapton - Rock n' Roll Hard
Nirvana - Lake of Fire
Nirvana - The Man who Sold the World
White Stripes - Seven Nation Army
The Doors - Roadhouse Blues

Larger Set (Beatles)
Ob la di ob la da
Across the Universe
A Day in the Life
A Hard Day’s Night
All I’ve Got To Do
And I Love Her
And Your Bird Can Sing
Any Time At All
Baby It’s You
Baby’s in Black
Because
Being for the Benefit of Mr. Kite
Birthday
Blue Jay Way
Can’t Buy Me Love
Come Together
Day Tripper
Dizzy Ms. Lizzie
Doctor Robert
Do You Want to Know a Secret?
Drive My Car
Eight Days a Week
Every Little Thing
Flying
From Me to You
Get Back
Good Day Sunshine
Got to Get You Into My Life
Hello Goodbye
Here Comes the Sun
Hey Bulldog
Hold Me Tight
Honey Don’t
I Call Your Name
I Don’t Want to Spoil the Party
I Feel Fine
I’ll Cry Instead
I’ll Follow the Sun
I’ll Get You
I’m Down
I’m Happy Just to Dance With You
I’m so tired

It Won't Be Long
I've Just Seen a Face
I Want to Hold Your Hand
I Will
Julia
Kansas City
Love Me Do
Lucy in the Sky with Diamonds
Magical Mystery Tour
Matchbox
Mean Mr.Mustard
Michelle
Money
Mr. Moonlight
Norwegian Wood (This Bird Has Flown)
Paperback Writer
Please Mister Postman
Please Please Me
P.S. I Love You
Revolution
Rock and Roll Music
Rocky Raccoon
Roll Over Beethoven
Sgt. Pepper's Lonely Hearts Club Band
She Loves You
She Said She Said
She's a Woman
Sun King
Tell Me What You See
Thank You Girl
The Ballad of John and Yoko
The Word
Things We Said Today
Think For Yourself
Till There Was You
Tomorrow Never Knows
Twist and Shout
We Can Work It Out
When I Get Home
When I'm Sixty-Four
With a Little Help from My Friends
Words of Love
Yellow Submarine
Yes It Is
Yesterday
You Like Me Too Much
You Really Got a Hold on Me

7.2 Source Code and Sample Midi Files

The source code that was produced during this guided research and some sample output files can be found online at : http://www.github.com/tpllaha/jammy

References

[1] Adam Alpern. Techniques for algorithmic composition of music. *On the web: http://hamp. hampshire. edu/˜ adaF92/algocomp/algocomp95. html*, 1995.

[2] Frederick P Brooks, AL Hopkins, Peter G Neumann, and WV Wright. An experiment in musical composition. *Electronic Computers, IRE Transactions on*, (3):175–182, 1957.

[3] Ella Gale, Oliver Matthews, Ben de Lacy Costello, and Andrew Adamatzky. Beyond markov chains, towards adaptive memristor network-based music generation. *arXiv preprint arXiv:1302.0785*, 2013.

[4] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

[5] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[6] Mantas LukošEvičIus and Herbert Jaeger. Survey: Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[7] Herbert Jaeger. *Short term memory in echo state networks*. GMD-Forschungszentrum Informationstechnik, 2001.

[8] Herbert Jaeger. The” echo state” approach to analysing and training recurrent neural networks-with an erratum note’. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148, 2001.

[9] Ian Simon, Dan Morris, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.

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

[11] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[12] Herbert Jaeger. scholarpedia. `http://www.scholarpedia.org/article/Echo_state_network`. Accessed: 2013-12-04.

[13] forumilab. midicsv,csvmidi. `http://www.fourmilab.ch/webtools/midicsv/`. Accessed: 2014-05-11.

[14] Roger N Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[15] Roger N Shepard. Geometrical approximations to the structure of musical pitch. *Psychological review*, 89(4):305, 1982.